PingDirectory™

Releases 7.0.1 & 7.0

Guides

PingDirectory™

Release 7.0.1

Server Administration Guide

Ping Identity®

Notice

PingDirectory™ Product Documentation

Trademarks

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Contents

Chapter 18: Managing Password Policies....................309

Chapter 19: Managing Replication....................325

Chapter 22: Managing Notifications and Alerts.. 425

Chapter 23: Managing the SCIM Servlet Extension....................................... 437

Chapter 1

Overview of the Server

Topics:

- *Server Features*
- *Administration Framework*
- *Server Tools Location*

The PingDirectory Server is a high performance, extensible directory and PingData Platform, written completely in Java™. The Directory Server centralizes consumer and user identity management information, subscriber data management, application configurations, and user credentials into a network, enterprise, or virtualized database environment. It provides seamless data management over a distributed system based on a standardized solution that meets the constant performance demands for today's markets. The Directory Server simplifies administration, reduces costs, and secures information in systems that scale for very large numbers of users.

This chapter provides an overview of the Directory Server's features and components.

Server Features

The PingDirectory Server is a powerful, 100% Java, production-proven PingData Platform solution for mission-critical and large-scale applications. The Directory Server provides an extensive feature-rich set of tools that can meet the production needs of your system.

- **Full LDAP Version 3 Implementation**. The Directory Server fully supports the Lightweight Directory Access Protocol version 3 (LDAP v3), which supports the Request For Comments (RFCs) specified in the protocol. The Directory Server provides a feature-rich solution that supports the core LDAPv3 protocol in addition to server-specific controls and extended operations.
- **High Availability**. The Directory Server supports N-way multi-master replication that eliminates single points of failure and ensures high availability for a networked topology. The Directory Server allows data to be stored across multiple machines and disk partitions for fast replication. The Directory Server also supports replication in entry-balancing proxy server deployments.
- **Administration Tools**. The Directory Server provides a full set of command-line tools, an Administrative Console, and a Java-based setup tool to configure, monitor, and manage any part of the server. The Directory Server has a task-based subsystem that provides automated scheduling of basic functions, such as backups, restores, imports, exports, restarts, and shutdowns. The set of utilities also includes a troubleshooting support tool that aggregates system metrics into a zip file, which administrators can send to your authorized support provider for analysis.
- **Self-Service Account Manager Application**. The Self Service Account Manager project, hosted at https://github.com/pingidentity/ssam, is a customizable web application allowing users to perform their own account registration, profile updates, and password changes. The project is for testing and development purposes, and is not a supported application.
- **Delegated Admin Application**. A Javascript-based web application can be installed for business users to manage identities stored in the Directory Server. The application provides delegated administration of identities for help desk or customer service representatives (CSR) initiating a password reset and unlock; an employee in HR updating an address stored within another employee profile; or an application administrator updating identity attributes or group membership to allow application SSO access.
- **Security Mechanisms**. The Directory Server provides extensive security mechanisms to protect data and prevent unauthorized access. Access control list (ACL) instructions are available down to the attribute value level and can be stored within each entry. The Directory Server allows connections over Secure Sockets Layer (SSL) through an encrypted communication tunnel. Clients can also use the StartTLS extended operation over standard, non-encrypted ports. Other security features include a privilege subsystem for fine-grained granting of rights, a password policy subsystem that allows configurable password validators and storage schemes, and SASL authentication mechanisms to secure data integrity, such as PLAIN, ANONYMOUS, EXTERNAL, CRAM-MD5, Digest-MD5, and GSSAPI. The Directory Server also supports various providers and mappers for certificate-based authentication in addition to the ability to encrypt specific entries or sensitive attributes. See the *PingDirectory Server Security Guide* for details.
- **Monitoring and Notifications**. The Directory Server supports monitoring entries using the PingDataMetrics Server, JConsole, Simple Network Management Protocol (SNMP), or using the Administrative Console. Administrators can track the response times for LDAP operations using a monitoring histogram as well as record performance statistics down to sub-second granularity. The Directory Server also supports configurable notifications, auditing, and logging subsystems with filtered logging capabilities.
- **Powerful LDAP SDK**. The Directory Server is based on a feature-rich LDAP SDK for Java, designed by Ping Identity. The Ping™ LDAP SDK is a Java API standard that overcomes the many limitations of the Java Naming and Directory Interface (JNDI) model. For example, JNDI does not address the use of LDAP controls and extended operations. The LDAP SDK for Java provides support for controls and extended operations to leverage the Ping Identity's extensible architecture for their applications.

  For more information on the LDAP SDK for Java, to go http://www.LDAP.com.
- **SCIM Extension**. The Directory Server provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure and fast manner.

- **Server SDK**. Ping Identity also provides the Server SDK, which is a library of Java packages, classes, and build tools to help in-house or third-party developers create client extensions for the PingDirectory Server, PingDirectoryProxy Server, and Data Sync Server. The servers were designed with a highly extensible and scalable architecture with multiple plug-in points for your customization needs. The Server SDK provides APIs to alter the behavior of each server's components without affecting its code base.

Administration Framework

The Directory Server provides an administration and configuration framework capable of managing stand-alone servers, server groups, and highly-available deployments that include multiple redundant server instances. Administrators can configure changes locally or remotely on a single server or on all servers in a server group. Each server configuration is stored as a flat file (LDIF) that can be accessed under the `cn=config` branch of the Directory Information Tree (DIT). Administrators can tune the configuration and perform maintenance functions over LDAP using a suite of command-line tools, or an Administrative Console (for configuration and monitoring). The Directory Server also provides plug-ins to extend the functionality of its components.

Server Tools Location

Ping Identity distributes the Directory Server, Administrative Console, and LDAP SDK for Java in zip format. After unzipping the file, you can access the `setup` utility in the server root directory, located at `PingDirectory`. The Directory Server stores a full set of command-line tools for maintaining your system in the `PingDirectory/bin` directory for UNIX® or Linux® machines and the `PingDirectory\bat` directory for Microsoft® Windows® machines.

Prior to installing the directory server, read Chapter 2 *Preparing Your Environment*, which presents important information on setting up your machines. Chapter 3 *Installing the Directory Server* presents procedures to install a server instance using the `setup` utility. This utility can be run in one of the two available installation modes: interactive command-line, and non-interactive command-line. Chapter 4 *Configuring the Directory Server* provides procedures to modify the configuration of a server instance or a group of servers using the command-line tools and the Administrative Console.

Chapter 2

Preparing Your Environment

Topics:

The PingDirectory Server offers a highly portable and scalable architecture that runs on multiple platforms and operating systems. The Directory Server is specifically optimized for those operating systems used in environments that process a very large number of entries.

This chapter presents some procedures to set up your server machines for optimal processing efficiency.

Before You Begin

The Directory Server requires certain software packages for the proper operation of the server. For optimized performance, the PingDirectory Server requires Java for 64-bit architectures. To view the minimum required Java version, access your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

It is also highly recommended that a Network Time Protocol (NTP) system be in place so that multi-server environments are synchronized and timestamps are accurate.

Installing Java

For optimized performance, the PingDirectory Server requires Java for 64-bit architectures. You can view the minimum required Java version on your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

Even if your system already has Java installed, you may want to create a separate Java installation for use by the PingDirectory Server to ensure that updates to the system-wide Java installation do not inadvertently impact the Directory Server. This setup requires that the JDK, rather than the JRE, for the 64-bit version, be downloaded.

To Install Java (Oracle/Sun)

1. Open a browser and navigate to the Oracle download site.
2. Download the latest version Java JDK. Click the JDK Download button corresponding to the latest Java update.
3. On the Java JDK page, click the Accept Licence Agreement button, then download the version based on your operating system.

Preparing the Operating System (Linux)

The PingDirectory Server has been extensively tested on multiple operating systems. We have found that several operating system optimizations lead to improved performance. These optimizations include increasing the file descriptor limit on Linux systems, setting filesystem flushes, editing OS-level environment variables, downloading some useful monitoring tools for Redhat Linux systems, and configuring for Huge Page support.

Configuring the File Descriptor Limits

The PingDirectory Server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. If needed, increase the file descriptor limit on the operating system.

If the operating system relies on `systemd`, refer to the Linux operating system documentation for instructions on setting the file descriptor limit.

To Set the File Descriptor Limit (Linux)

The Directory Server allows for an unlimited number of connections by default but is restricted by the file descriptor limit on the operating system. Many Linux distributions have a default file descriptor limit of 1024 per process, which may be too low for the server if it needs to handle a large number of concurrent connections.

Once the operating system limit is set, the number of file descriptors that the server will use can be configured by either using a `NUM_FILE_DESCRIPTORS` environment variable, or by creating a `config/num-file-descriptors` file with a single line such as, `NUM_FILE_DESCRIPTORS=12345`. If these are not set, the default of 65535 is used. This is strictly optional if wanting to ensure that the server shuts down safely prior to reaching the file descriptor limit.

1. Display the current hard limit of your system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

   ```
   ulimit -aH
   ```

2. Edit the `/etc/sysctl.conf` file. If there is a line that sets the value of the `fs.file-max` property, make sure its value is set to at least 65535. If there is no line that sets a value for this property, add the following to the end of the file:

   ```
   fs.file-max = 65535
   ```

3. Edit the `/etc/security/limits.conf` file. If the file has lines that sets the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before "#End of file"). Also note that you should insert a tab, rather than spaces, between the columns.

   ```
   * soft nofile 65535
   * hard nofile 65535
   ```

4. Reboot your system, and then use the `ulimit` command to verify that the file descriptor limit is set to 65535.

   ```
   # ulimit -n
   ```

Note: For RedHat 7 or later, modify the `20-nproc.conf` file to set both the open files and max user processes limits:

```
/etc/security/limits.d/20-nproc.conf

 Add or edit the following lines if they do not already exist:

 *          soft    nproc     65536
 *          soft    nofile    65536
 *          hard    nproc     65536
 *          hard    nofile    65536
 root       soft    nproc     unlimited
```

File System Tuning

Newer ext4 systems use delayed allocation to improve performance. This delays block allocation until it writes data to disk. Delayed allocation improves performance and reduces fragmentation by using the actual file size to improve block allocation. This feature may cause a risk of data loss in cases where a system loses power before all of the data has been written to disk. This may occur if a program is replacing the contents of a file without forcing a write to the disk with `fsync`. Make sure the default `auto_da_alloc` option is enabled on ext4 filesystems.

Administrators can tune ext3 and ext4 filesystems by setting the filesystem flushes and noatime to improve server performance. The following changes can be made in the `/etc/fstab` file.

To Set the Filesystem Flushes

With the out-of-the-box settings on Linux systems running the ext3 filesystem, the data is only flushed to disk every five seconds. If the Directory Server is running on a Linux system using the `ext3` filesystem, consider editing the mount options for that filesystem to include the following:

```
commit=1
```

This variable changes the flush frequency from five seconds to one second.

You should also set the flush frequency to the `/etc/fstab` file. Doing the change via the mount command alone will not survive across reboots.

To Set noatime on ext3 and ext 4 Systems

If you are using an `ext3` or `ext4` filesystem, it is recommended that you set `noatime`, which turns off any *atime* updates during read accesses to improve performance. You should also set the flush frequency to the `/etc/fstab` file. Doing the change via the mount command alone will not survive across reboots.

- Run the following command on an ext3 system.

```
# mount -t ext3 -o noatime /dev/fs1
```

- Run the following command on an ext34 system.

```
# mount -t ext4 -o noatime /dev/fs1
```

Setting the Maximum User Processes

On some Linux distributions (Redhat Enterprise Linux Server/CentOS 6.0 or later), the default maximum number of user processes is set to 1024, which is considerably lower than the same parameter on older distributions (e.g., RHEL/CentOS 5.x). The default value of 1024 leads to some JVM memory errors when running multiple servers on a machine due to each Linux thread being counted as a user process.

At startup, the Directory Server and its tools automatically attempt to raise the maximum user processes limit to 16,383 if the value reported by `ulimit` is less than that. If, for any reason, the server is unable to automatically set the maximum processes limit to 16,383, an error message will be displayed. It is recommended that the limit be set explicitly in `/etc/security/limit.conf`. For example:

```
* soft nproc 65535
* hard nproc 65535
```

The (*) can be replaced with the name of the user under which the software will run. These settings can also be manually configured by setting the `NUM_USER_PROCESSES` environment variable to 16383 or by setting the same variable in a file named `config/num-user-processes`.

About Editing OS-Level Environment Variables

Certain environment variables can impact the Directory Server in unexpected ways. This is particularly true for environment variables that are used by the underlying operating system to control how it uses non-default libraries.

For this reason, the Directory Server explicitly overrides the values of key environment variables like *PATH*, *LD_LIBRARY_PATH*, and *LD_PRELOAD* to ensure that something set in the environments that are used to start the server does not inadvertently impact its behavior.

If there is a legitimate need to edit any of these environment variables, the values of those variables should be set by manually editing the `set_environment_vars` function of the `lib/_script-util.sh` script. You will need to stop (bin/stop-server) and re-start (bin/start-server) the server for the change to take effect.

Install sysstat and pstack (Red Hat)

For Red Hat® Linux systems, you should install a couple of packages, `sysstat` and `pstack`, that are disabled by default, but are useful for troubleshooting purposes in the event that a problem occurs. The troubleshooting tool `collect-support-data` uses the `iostat`, `mpstat`, and `pstack` utilities to collect monitoring, performance statistics, and stack trace information on the server's processes. For Red Hat systems, make sure that these packages are installed, for example:

```
$ sudo yum install sysstat gdb dstat -y
```

Install dstat (SUSE Linux)

The `dstat` utility is used by the `collect-support-data` tool and can be obtained from the OpenSuSE project website. The following example shows how to install the `dstat` utility on SuSE Enterprise Linux 11 SP2:

1. Login as Root.
2. Add the appropriate repository using the `zypper` tool.
3. Install the `dstat` utility.

```
$ zypper install dstat
```

Disable Filesystem Swapping

It is recommended that any performance tuning services like `tuned` be disabled. If performance tuning is required, `vm.swappiness` can be set by cloning the existing performance profile then adding `vm.swappiness = 0` to the new profile's `tuned.conf` file. This file is located at `/usr/lib/tuned/profile-name/tuned.conf`. The updated profile is then selected by running `tuned-adm profile customized_profile`.

Omit vm.overcommit_memory

Administrators should be aware that an improperly configured value for the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file can cause the `setup` or `start-server` tool to fail.

For Linux systems, the `vm.overcommit_memory` property sets the kernel policy for memory allocations. The default value of 0 indicates that the kernel determines the amount of free memory to grant a `malloc` call from an application. If the property is set to a value other than zero, it could lead the operating system to grab too much memory, depriving memory for the `setup` or `start-server` tool.

We recommend omitting the property in the `/etc/sysctl.conf` file to ensure that enough memory is available for these tools.

Managing System Entropy

Entropy is used to calculate random data that is used by the system in cryptographic operations. Some environments with low entropy may have intermittent performance issues with SSL-based communication. This is more typical on virtual machines, but can occur in physical instances as well. Monitor the `kernel.random.entropy_avail` in `sysctl` value for best results.

If necessary, update `$JAVA_HOME/jre/lib/security/java.security` to use `file:/dev/./urandom` for the `securerandom.source` property.

Running as a Non-Root User (Linux)

To run as a non-root user but still allow connections on a privileged port, two options are available:

- **Use a Load-Balancer or Directory Proxy Server**. In many environments, the server can be run on a non-privileged port but can be hidden by a hardware load-balancer or LDAP Directory Proxy Server.
- **Use `netfilter`**. The `netfilter` mechanism, exposed through the `iptables` command, can be used to automatically redirect any requests from a privileged port to the unprivileged port on which the server is listening.

Enabling the Server to Listen on Privileged Ports (Linux)

Linux systems have a mechanism called capabilities that is used to grant specific commands the ability to do things that are normally only allowed for a root account. It may be convenient to enable the server to listen on privileged ports while running as a non-root user.

The `setcap` command is used to assign capabilities to an application. The `cap_net_bind_service` capability enables a service to bind a socket to privileged ports (port numbers less than 1024). If Java is installed in `/ds/java` (and the Java command to run the server is `/ds/java/bin/java`), the Java binary can be granted the `cap_net_bind_service` capability with the following command:

```
$ sudo setcap cap_net_bind_service=+eip /ds/java/bin/java
```

The java binary needs an additional shared library (`libjli.so`) as part of the Java installation. More strict limitations are imposed on where the operating system will look for shared libraries to load for commands that have capabilities assigned. So it is also necessary to tell the operating system where to look for this library. This can be done by creating the file `/etc/ld.so.conf.d/libjli.conf` with the path to the directory that contains the `libjli.so` file. For example, if the Java installation is in `/ds/java`, the contents of that file should be:

```
/ds/java/lib/amd64/jli
```

Run the following command for the change to take effect:

```
$ sudo ldconfig -v
```

Chapter

3

Installing the Server

Topics:

After you have prepared your hardware and software system based on the instructions in Chapter 2, you can begin the setup process using of the PingDirectory Server's easy-to-use installation modes.

This chapter presents the various installation options and procedures available to the administrator.

Getting the Installation Packages

To begin the installation process, obtain the latest ZIP release bundle from Ping Identity and unpack it in a folder of your choice. The release bundle contains the Directory Server code, tools, and package documentation.

To Unpack the Build Distribution

1. Download the latest zip distribution of the Directory Server software.
2. Unzip the compressed zip archive file in a directory of your choice.

   ```
   $ unzip PingDirectory-<version>.zip
   ```

   You can now set up the Directory Server.

About the RPM Package

PingDirectory Server supports the PingDirectory Server release bundle in an RPM Package Manager (RPM) package for customers who require it. By default, the RPM unpacks the code at /opt/ping-identity/ds/PingDirectory, after which you can run the `setup` command to install the server at that location.

If the RPM install fails for any reason, you can perform an RPM erase if the RPM database entry was created and manually remove the target RPM install directory (e.g., “/opt/ping-identity/ds/PingDirectory” by default). You can install the package again once the system is ready.

To Install the RPM Package

1. Download the latest RPM distribution of the Directory Server software.
2. Unpack the build using the `rpm` command with the --install option. By default, the build is unpacked to `/opt/ping-identity/ds/PingDirectory`. If you want to place the build at another location, use the `--prefix` option and specify the file path of your choice.

   ```
   $ rpm --install PingDirectory-<version>.rpm
   ```
3. From /opt/ping-identity/ds/PingDirectory/PingDirectory, run the `setup` command to install the server on the machine.

About the Layout of the Directory Server Folders

Once you have unzipped the Directory Server distribution file, you will see the following folders and command-line utilities, shown in the table below.

Table 1: Layout of the Directory Server Folders

Directories/Files/Tools	Description
License.txt	Licensing agreement for the Directory Server.
README	README file that describes the steps to set up and start the Directory Server.
bak	Stores the physical backup files used with the `backup` command-line tool.
bat	Stores Windows-based command-line tools for the Directory Server.
bin	Stores UNIX/Linux-based command-line tools for the Directory Server.
classes	Stores any external classes for server extensions.
collector	Used by the server to make monitored statistics available to the PingDataMetrics Server.

Directories/Files/Tools	Description
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
db	Stores the Oracle Berkeley Java Edition database files for the Directory Server.
docs	Provides the product documentation.
import-tmp	Stores temporary imported items.
ldif	Stores any LDIF files that you may have created or imported.
legal-notices	Stores any legal notices for dependent software used with the Directory Server.
lib	Stores any scripts, jar, and library files needed for the server and its extensions.
locks	Stores any lock files in the backends.
logs	Stores log files for the Directory Server.
metrics	Stores the metrics that can be gathered for this server and surfaced in the PingDataMetrics Server.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
revert-update	The `revert-update` tool for UNIX/Linux systems.
revert-update.bat	The `revert-update` tool for Windows systems.
setup	The `setup` tool for UNIX/Linux systems.
setup.bat	The `setup` tool for Windows systems.
scim-data-tmp	Used to create temporary files containing SCIM request data.
uninstall	The `uninstall` tool for UNIX/Linux systems.
uninstall.bat	The `uninstall` tool for Windows systems.
update	The `update` tool for UNIX/Linux systems.
update.bat	The `update` tool for Windows systems.
Velocity	Stores any customized Velocity templates and other artifacts (CSS, Javascript, images), or Velocity applications hosted by the server.

About the Server Installation Modes

One of the strengths of the PingDirectory Server is the ease with which you can install a server instance using the `setup` tool. The `setup` tool allows you to quickly install and configure a stand-alone Directory Server instance.

To install a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode prompts for information during the installation process. To run the installation in this mode, use the `setup --cli` command.
- **Non-Interactive Command-Line Mode**. Non-interactive command-line mode is designed for setup scripts to automate installations or for command-line usage. To run the installation in this mode, `setup` must be run with the --no-prompt option as well as the other arguments required to define the appropriate initial configuration.

All installation and configuration steps should be performed while logged on to the system as the user or role under which the Directory Server will run.

Before You Begin

After you have unzipped the Directory Server ZIP file, you may want to carry out the following functions depending on your deployment requirements:

- **Custom Schema Elements**. If your deployment uses custom schema elements in a custom schema file (for example, `98-schema.ldif`), you may do one of the following:
  - Copy your custom schema file to the `config/schema` directory before running setup.
  - Copy your custom schema file to the `config/schema` directory after setup and re-start the server. If replication is enabled, the restart will result in the schema replicating to other servers in the replication topology.
  - Use the **Schema Editor** after setup. If replication is enabled, schema definitions added through the **Schema Editor** will replicate to all servers in the replication topology without the need for a server restart.
- **Certificates**. If you are setting up a new machine instance, copy your keystore and truststore files to the `<server-root>/config` directory prior to running setup. The keystore and truststore passwords can be placed, in clear text, in corresponding `keystore.pin` and `truststore.pin` files in `<server-root>/config`.
- **Encryption Passphrase**. Encryption for directory data, backups, LDIF exports, and log files can be enabled during setup by providing or generating an encryption key with a passphrase.
- **Locations**. Location names are used to define a grouping of PingDirectory Server products based on physical proximity. For example, a location is most often associated with a single datacenter location. During the installation, assign a location to each server for optimal inter-server behavior. The location assigned to a server within Global Configuration can be referenced by components within the server as well as processes external to the server to satisfy "local" versus "remote" decisions used in replication, load balancing, and failover.
- **Validate ACIs**. Many directory servers allow for less restrictive application of its access control instructions (ACIs), so that they accept invalid ACIs. For example, if a Sun/Oracle server encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the PingDirectory Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a PingDirectory Server. If you are migrating from a Sun/Oracle deployment to a PingDirectory Server, the PingDirectory Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. For more information, see *Validating ACIs Before Migrating Data*.

Important:

Each Server Deployment Requires an Execution of Setup - Duplicating a Server-root is not Supported. The installation of the server does not write or require any data outside of the server-root directory. After executing `setup`, copying the server-root to another location or system, in order to *duplicate* the installation, is not a supported method of deployment. The server-root can be moved to another host or disk location if a host or file system change is needed.

PingDirectory Server License Keys

License keys are required to install all PingDirectory Server products. Obtain licenses through Salesforce or from https://www.pingidentity.com/en/account/request-license-key.html.

- A license is always required for setting up a new single server instance and can be used site-wide for all servers in an environment. When cloning a server instance with a valid license, no new license is needed.
- A new license must be obtained when updating a server to a new major version, for example from 6.2 to 7.0. Licenses with no expiration date are valid until the server is upgraded to the next major version. A prompt for a new license is displayed during the update process.
- A license may expire on particular date. If a license does expire, obtain a new license and install it using `dsconfig` or the Administrative Console. The server will provide a notification as the expiration date approaches. License details are available using the server's `status` tool.

When installing the server, specify the license key file in one of the following ways:

- Copy the license key file to the server root directory before running setup. The interactive `setup` tool will discover the file and not require input. If the file is not in the server root, the `setup` tool will prompt for its location.
- If the license key is not in the server root directory, specify the `--licenseKeyFile` option for non-interactive setup, and the path to the file.

Setting Up the Directory Server in Interactive Mode

The `setup` tool also provides an interactive text-based command-line interface to set up a Directory Server instance.

To Install the Directory Server in Interactive Mode

1. Unzip the distribution ZIP file, review *Before You Begin*, and then go to the server root directory. Use the `setup` utility with the --cli option to install the server in interactive mode.

   ```
   $ ./setup
   ```

   If the *JAVA_HOME* environment variable is set to an older version of Java, explicitly specify the path to the Java JDK installation during the setup process. Either set the *JAVA_HOME* environment variable with the Java JDK path or execute the `setup` command in a modified Java environment using the `env` command.

   ```
   $ env JAVA_HOME=/ds/java ./setup
   ```
2. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
3. Enter the fully qualified host name or IP address of the local host, or press **Enter** to accept the default.
4. Enter the root user DN, or press **Enter** to accept the default (cn=Directory Manager).
5. Enter and confirm the root user password.
6. Press **Enter** to enable the Ping Identity services (Configuration, Consent, Delegated Admin, Documentation, and SCIM) and Administrative Console over HTTPS. After setup, individual services and applications can be enabled or disabled by configuring the HTTPS Connection Handler.
7. Enter the port on which the Directory Server will accept connections from HTTPS clients, or press **Enter** to accept the default.
8. Enter the port on which the Directory Server will accept connections from LDAP clients, or press **Enter** to accept the default.
9. The next two options enable using LDAPS or StartTLS. Type `no` to use a standard LDAP connection, or accept the default (`yes`) to enable both. Enabling LDAPS configures the LDAPS Connection Handler to allow SSL over its client connections. Enabling StartTLS configures the LDAP Connection Handler to allow StartTLS.
10. Select the certificate option for this server:

    - Generate a self-signed certificate for testing purposes only.
    - To use an existing certificate using a Java Keystore, enter the keystore path and keystore PIN.
    - To use an existing certificate using use a PKCS#12 keystore, enter the keystore path and the keystore PIN.
    - To use the PKCS#11 token, enter only the keystore PIN.
11. Choose the desired encryption for the directory data, backups, and log files from the choices provided:

    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.

- Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
- Do not encrypt server data.

12. Type the base DN for the data, or accept the default base DN of `dc=example,dc=com`.
13. Choose an option to generate and import sample data. Type the desired number of entries, or press `Enter` to accept the default number (10000). This option is used for quick evaluation of the Directory Server.

    See *Initializing Data onto the Server* if you want to use other options to initialize the server.
14. Choose the option to tune the amount of memory that will be consumed by the Directory Server and its tools.
15. Press `Enter` to prime or preload the database cache at startup prior to accepting client connections.

    Priming the cache can increase the startup time for the Directory Server but provides optimum performance once startup has completed. This option is best used for strict throughput or response time performance requirements, or if other replicas in a replication topology can accept traffic while this Directory Server instance is starting. Priming the cache also helps determine the recommended JVM option, CMSInitiatingOccupancyFraction, when a Java garbage collection pause occurs. See *JVM Garbage Collection Using CMS*.
16. Enter a location name for this server.
17. Enter a unique instance name for this server. Once set, the name cannot be changed.
18. Press **Enter** to accept the default (yes) to start the Directory Server after the configuration has completed. To configure additional settings or import data, type `no` to keep the server in shutdown mode.
19. Choose an option to continue server set up.
20. On the **Setup Summary** page, confirm the configuration. Press **Enter** to accept the default (set up with the parameters given), enter the option to repeat the installation process, or enter the option to cancel the setup completely.

Installing the Directory Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. Non-interactive mode is useful when setting up production or QA servers with specific configuration requirements.

The non-interactive command-line mode requires that all mandatory options be present for each command call. If there are missing or incorrect arguments, the `setup` tool fails and aborts the process. You must also use a --no-prompt option to suppress interactive output, except for errors, when running in non-interactive mode. Additionally, you must also use the --acceptLicense option and specify the port using the --ldapPort or --ldapsPort option. If neither option is specified, an error message is displayed. To view the license, run bin/review-license command.

To automatically tune the JVM to use maximum memory, use the --aggressiveJVMTuning and --maxHeapSize {memory} options. To preload the database at startup, use the --primeDB option.

To configure a deployment using a truststore, see *Installing the Directory Server in Non-Interactive Mode with a Truststore*.

To see a description of the available command-line options for the `setup` tool, use `setup --help`.

To Install the Directory Server in Non-Interactive Mode

The following procedure shows how to install a Directory Server in a production or QA environment with no security enabled.

- Unzip the distribution ZIP file, review “Before You Begin”, and then use `setup` with the --cli and --no-prompt options for non-interactive mode from the `<server-root>` directory. The following command uses the default root user DN (cn=Directory Manager) with the specified --rootUserPassword option. You must include the --acceptLicense option or the setup will generate an error message.

```
$ ./setup --cli --no-prompt --rootUserPassword "password" \
  --baseDN "dc=example,dc=com" --acceptLicense --ldapPort 389
```

To Install the Directory Server in Non-Interactive Mode with a Truststore

You can set up the Directory Server using an existing truststore for secure communication. This section assumes that you have an existing keystore and truststore with trusted certificates.

- Unzip the distribution ZIP file, review *Before You Begin*, and then, from the server root directory, use `setup` with the --cli and --no-prompt options for non-interactive mode. The following example enables security using both SSL and StartTLS. It also specifies a JKS keystore and truststore that define the server certificate and trusted CA. The `userRoot` database contents will remain empty and the base DN entry will not be created.

  ```
  $ ./setup --cli --no-prompt --rootUserPassword "password" \
    --baseDN "dc=example,dc=com" --ldapPort 389 --enableStartTLS \
    --ldapsPort 636 --useJavaKeystore config/keystore.jks \
    --keyStorePasswordFile config/keystore.pin \
    --certNickName server-cert --useJavaTrustStore config/truststore.jks \
    --acceptLicense
  ```

  The password to the private key with the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined with the Administrative Console or the `dsconfig` tool by editing the Trust Manager Provider standard configuration object.

Installing a Lightweight Server

Users who want to demo or test a lightweight version of the Directory Server on a memory-restricted machine can do so by removing all unused or unneeded configuration objects. All configuration entries, whether enabled or not, take up some amount of memory to hold the definition and listeners that will be notified of changes to those objects.

The configuration framework will not allow you to remove objects that are referenced, and in some cases if you have one configuration object referencing another but really do not need it, then you will first need to remove the reference to it. If you try to remove a configuration object that is referenced, both `dsconfig` and the Administrative Console should prevent you from removing it and will tell you what still references it.

Depending on your test configuration, some example configuration changes that can be made are as follows:

- **Reduce the number of worker threads**. Each thread has a stack associated with it, and that consumes memory. If you're running a bare-bones server, then you probably do not have enough load to require a lot of worker threads.

  ```
  $ bin/dsconfig set-work-queue-prop \
    --set num-worker-threads:8 \
    --set num-administrative-session-worker-threads:4 \
    --set max-work-queue-capacity:100
  ```
- **Reduce the percentage of JVM memory used for the JE database cache**. When you have a memory-constrained environment, you want to ensure that as much of the memory that is there is available for use during processing and not tied up caching database contents.

  ```
  $ bin/dsconfig set-backend-prop --backend-name userRoot --set db-cache-
  percent:5
  ```
- **Disable the Dictionary Password Validator**. The Dictionary Password Validator takes a lot of memory to hold its dictionary. Disabling it will free up some memory. You can delete the other password validators if not needed, such as Attribute Value, Character Set, Length-based, Repeated Characters, Similarity-based, or Unique Characters Password Validator.

  ```
  $ bin/dsconfig delete-password-validator --validator-name Dictionary
  ```
- **Remove non-essential schema files**. Although not recommended for production deployments, some candidates that you can remove are the following: `03-rfc2713.ldif`, `03-rfc2714.ldif`, `03-rfc2739.ldif`, `03-rfc2926.ldif`, `03-rfc2985.ldif`, `03-rfc3712.ldif`, `03-uddiv3.ldif`.

There are other items that can be removed, depending on your desired configuration. Contact your authorized support provider for assistance.

Running the Status Tool

The Directory Server provides a `status` tool that outputs the current state of the server as well as other information, such as server version, JE Environment statistics, Operation Processing Times, Work Queue, and Administrative Alerts. The `status` tool is located in the `bin` directory (UNIX, Linux) or the `bat` directory (Windows).

To Run the Status Tool

- Run the `status` command on the command line. The following command displays the current Directory Server status and limits the number of viewable alerts in the last 48 hours. It provides the current state of each connection handler, data sources, JE environment statistics, processing times by operation type and current state of the work queue.

```
$ bin/status --bindDN “uid=admin,dc=example,dc=com” --bindPassword secret

          --- Server Status ---
Server Run Status:    Started 28/Mar/2012:10:47:17.000 -0500
Operational Status:   Available
Open Connections:     13
Max Connections:      13
Total Connections:    50

          --- Server Details ---
Host Name:            server1.example.com
Administrative Users: cn=Directory Manager
Installation Path:    PingDirectory
Server Version:       PingDirectory Server 7.0.1.0
Java Version:         jdk-7u9

          --- Connection Handlers ---
Address:Port : Protocol : State
-------------:----------:---------
0.0.0.0:1389 : LDAP     : Enabled
0.0.0.0:1689 : JMX      : Disabled
0.0.0.0:636  : LDAPS    : Disabled

           --- Data Sources ---
Base DN:             dc=example,dc=com
Backend ID:          userRoot
Entries:             2003
Replication:         Enable
Replication Backlog: 0
Age of Oldest Backlog Change: not available

           --- JE Environment ---
ID                 : Cache Full :  Cache   : On-Disk : Alert
-------------------:------------:---------:----------:------
replicationChanges : 6 %     : 328.8 kb : 30.4 kb : None
userRoot        : 9 %        : 6.2mb    : 146.6mb : None

           --- Operation Processing Time ---
Op Type   : Total Ops : Avg Resp Time (ms)
----------:-----------:------------------
Add       : 0         : 0.0
Bind      : 0         : 0.0
Compare   : 0         : 0.0
Delete    : 0         : 0.0
Modify    : 2788567   : 0.921
Modify    : 0         : 0
DN        : 2267266   : 0.242
Search    : 5055833   : 0.616
```

```
All

            --- Work Queue ---
           : Recent : Average : Maximum
-----------:--------:---------:--------
Queue Size : 4   : 0    : 10
% Busy  : 26  : 5    : 100

            --- Administrative Alerts ---
Severity : Time                       : Message
---------:----------------------------:---------------------------------------------
Info     : 28/Mar/2012 10:47:17 -0500 : The Directory Server has started
 successfully
Info     : 28/Mar/2012 10:47:14 -0500 : The Directory Server is starting
Info     : 28/Mar/2012 10:44:22 -0500 : The Directory Server has started
 successfully
Info     : 28/Mar/2012 10:44:18 -0500 : The Directory Server is starting

Shown are alerts of type [Info,Warning,Error,Fatal] from the past 48 hours
 Use the
--maxAlerts and/or --severity options to filter this list
```

Note: By default, the `status` command displays the alerts generated in the last 48 hours. You can limit the number of alerts by using the --maxAlerts option.

Where To Go From Here

After you have set up your Directory Server instance, you can configure any specific server settings, import your user database, or run initial performance tests to optimize your server’s throughput.

- **Log into the Administrative Console**. Become familiar with configuration options through the Administrative Console interface. The URL is based on the hostname and HTTPS port specified during installation, such as https://hostname.com:443/console.
- **Apply Server Configurations**. Apply your server configuration changes individually or using a `dsconfig` batch file. The batch file defines the Directory Server configuration tool, `dsconfig`, commands necessary to configure your server instance. For more information on using batch files, see *Using dsconfig in Batch Mode*.

  If you are migrating from a Sun Java System 5.x, 6.x, 7.x directory server, you can use the `bin/migrate-sun-ds-config` command to migrate your configuration settings to this newly installed server instance.
- **Import Data**. Import user data using the `import-ldif` tool. The import serves as an initial test of the schema settings.

  ```
  $ bin/import-ldif --backendID userRoot --ldifFile ../user-data.ldif
  ```
- **Install and Configure the Delegated Admin Application**. A Javascript-based web application can be installed for business users to manage identities stored in the Directory Server. The application provides delegated administration of identities for help desk or customer service representatives (CSR) initiating a password reset and unlock; an employee in HR updating an address stored within another employee profile; or an application administrator updating identity attributes or group membership to allow application SSO access.
- **Run Performance Tests**. The Directory Server provides two tools for functional performance testing using in-house LDAP clients that accesses the server directly: `searchrate` (tests search performance) and `modrate` (tests modification performance):

  ```
  $ bin/searchrate --baseDN "dc=example,dc=com" --scope sub \
    --filter "(uid=user.[0-1999])" --attribute givenName --attribute sn \
    --attribute mail --numThreads 10

  $ bin/modrate --entryDN "uid=user.[0-1999],ou=People,dc=example,dc=com" \
    --attribute description --valueLength 12 --numThreads 10
  ```

Working with Multiple Backends

You can create multiple local database backends, each containing one or more different base DNs. There should be at most one replicating domain on each local database backend. The replication domain should not span multiple local database backends. The typical entry-balancing configuration involves two local database backends: one backend to serve the global domain data that resides above the entry-balancing point and a backend that is defined with the entry-balancing point as the base DN, such as `ou=people,dc=example,dc=com`.

With multiple local database backends configured, the data existing with each backend can be managed independently. In addition, separate index settings are applied to each local database backend.

When creating multiple databse backends, consider the following:

- No two backends may have the same base DN.
- If any base DN for a given backend is subordinate to a base DN on another backend, then all base DNs on that backend must be subordinate to the base DN of the other backend.
- The total of all db-cache-percent values should be no more than 65-70% in most cases and should never be configured to exceed 100%.

Importing Data

After installation, the database, such as userRoot, will need to have data imported. For a server to be added to a replicating set, the database will be imported as part of the `dsreplication initialize` operation, which is performed after `dsreplication enable`. A server that will not be added to a replicating set, or the first server of a future replicating set, should have data imported with the `bin/import-ldif` tool. See Chapter 8, Importing and Exporting Data, for more infomation about the `bin/import-ldif` tool.

Generating Sample Data

The PingDirectory Server provides LDIF templates that can be used to generate sample entries to initialize your server. You can generate the sample data with the `make-ldif` utility together with template files that come bundled with the ZIP build, or you can use templates files that you create yourself. The templates create sequential entries that are convenient for testing the PingDirectory Server with a range of dataset sizes. The Directory Server templates are located in the `config/MakeLDIF`.

To randomize the data, the `make-ldif` command has a --randomSeed option that can be used to seed the random number generator. If this option is used with the same seed value, the template will always generate exactly the same LDIF file.

The sample data templates generate a dataset with basic access control privileges that grants anonymous read access to anyone, grants users the ability to modify their own accounts, and grants the Admin account full privileges. The templates also include the uid=admin and ou=People entries necessary for a complete dataset. You can bypass the `make-ldif` command entirely and use the --templateFile option with the `import-ldif` tool.

- Use the `make-ldif` command to generate sample data. The command generates 10,000 sample entries and writes them to an output file, `data.ldif`. The random seed generator is set to 0.

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-10k.template \
  --ldifFile /path/to/data.ldif --randomSeed 0
```

To Import Data on the Directory Server Using Offline Import

1. Create an LDIF file that contains entries, or locate an existing file.

   The import-ldif tool requires an LDIF file, which conforms to standard LDIF syntax without change records. This means the changeType attribute is not allowed in the input LDIF. For information on adding entries to the Directory Server, see Managing Entries.
2. Stop the Directory Server.

3. Use the offline `import-ldif` to import data from an LDIF file to the Directory Server. For assistance with the list of options, run `import-ldif --help`.

   In the following example, the data is imported from the `data.ldif` file to the `userRoot` backend. If any entry is rejected due to a schema violation, then the entry and the reason for the rejection is written to the `rejects.ldif` file. Skipped entries, written to `skipped.ldif`, occur if an entry cannot be placed under a branch node in the DIT or if exclusion filtering is used (`--excludeBranch`, `--excludeAttribute`, or `--excludeFilter`). The `--overwrite` option instructs `import-ldif` to overwrite existing skipped and rejected files. The `--overwriteExistingEntries` option indicates that any existing data in the backend should be overwritten. Finally, the `--stripTrailingSpaces` option strips trailing spaces on attributes that would otherwise result in a LDIF parsing error.

   ```
   $ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif --
   rejectFile
       rejects.ldif --skipFile skipped.ldif --overwrite --
   overwriteExistingEntries --stripTrailingSpaces
   ```
4. Re-start the Directory Server.

Running the Server

To start the Directory Server, run the bin/start-server command on UNIX or Linux systems (an analogous command is in the `bat` folder on Microsoft Windows systems). The bin/start-server command starts the Directory Server as a background process when no options are specified. To run the Directory Server as a foreground process, use the bin/start-server command with the --nodetach option.

To Start the Directory Server

Use bin/start-server to start the server.

```
$ bin/start-server
```

To Run the Server as a Foreground Process

1. Enter bin/start-server with the --nodetach option to launch the Directory Server as a foreground process.

   ```
   $ bin/start-server --nodetach
   ```
2. You can stop the Directory Server by pressing `CNTRL+C` in the terminal window where the server is running or by running the bin/stop-server command from another window.

To Start the Server at Boot Time

By default, the PingDirectory Server does not start automatically when the system is booted. Instead, you must manually start it with the bin/start-server command. To configure the Directory Server to start automatically when the system boots, use the `create-systemd-script` utility to create a script, or create the script manually.

1. Create the service unit configuration file in a temporary location where "ds" is the user the PingDirectory will run as.

   ```
   $ bin/create-systemd-script \
        --outputFile /tmp/ping-directory.service \
        --userName ds
   ```
2. As a root user, copy the `ping-directory.service` configuration file into the `/etc/systemd/system` directory.
3. Reload `systemd` to read the new configuration file.

   ```
   $ systemctl daemon-reload
   ```
4. To start the PingDirectory, use the `start` command.

```
$ systemctl start ping-directory.service
```

5. To configure the PingDirectory to start automatically when the system boots, use the `enable` command.

   ```
   $ systemctl enable ping-directory.service
   ```

6. Log out as root.

   If on an RC system, this task is done by creating the startup script with `bin/create-rc-script` and moving it to the `/etc/init.d` directory. Create symlinks to it from the `/etc/rc3.d` directory (staring with an "S" to ensure that the server is started) and `/etc/rc0.d` directory (starting with a "K" to ensure that the server is stopped).

Logging into the Administrative Console

After the server is installed, access the Administrative Console, `https://hostname:HTTPport/console/login`, to verify the configuration and manage the server. To log into the Administrative Console, use the initial root user DN specified during setup (by default `cn=Directory Manager`).

The `dsconfig` command or the Administrative Console can be used to create additional root DN users in `cn=Root DNs,cn=config`. These new users require the fully qualified DN as the login name, such as `cn=new-admin,cn=Root DNs,cn=config`. To use a simple user name (with out the `cn=` prefix) for logging into the Administrative Console, the root DN user must have the `alternate-bind-dn` attribute configured with an alternate name, such as "admin."

By default the link to the Administrative Console is `https://hostname:HTTPport/console/login`.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package (`/server-root/resource/admin-console.zip`) can be installed according to that container's documentation.

Stopping the Directory Server

The Directory Server provides a simple shutdown script, bin/stop-server, to stop the server. You can run it manually from the command line or within a script.

If the Directory Server has been configured to use a large amount of memory, then it can take several seconds for the operating system to fully release the memory and make it available again. If you try to start the server too quickly after shutting it down, then the server can fail because the system does not yet have enough free memory. On UNIX systems, run the `vmstat` command and watch the values in the "free" column increase until all memory held by the Directory Server is released back to the system.

You can also set a configuration option that specifies the maximum shutdown time a process may take.

To Stop the Server

- Use the bin/stop-server tool to shut down the server.

  ```
  $ bin/stop-server
  ```

To Schedule a Server Shutdown

- Use the `bin/stop-server` tool with the --stopTime YYYYMMDDhhmmss option to schedule a server shutdown.

  The Directory Server schedules the shutdown and sends a notification to the `server.out` log file. The following example sets up a shutdown task that is scheduled to be processed on June 6, 2012 at 8:45 A.M. CDT. The server uses the UTC time format if the provided timestamp includes a trailing "Z", for example, 20120606134500Z. The command also uses the --stopReason option that writes the reason for the shut down to the logs.

  ```
  $ bin/stop-server --stopTime 20120606134500Z --port 1389 \
  ```

```
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --stopReason "Scheduled offline maintenance"
```

To Restart the Server

Re-start the Directory Server using the bin/stop-server command with the `--restart` or `-R` option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again.

- Go to the server root directory, and run the bin/stop-server command with the -R or --restart options.

```
$ bin/stop-server --restart
```

Run the Server as a Microsoft Windows Service

The server can run as a Windows service on Windows Server 2012 R2 and Windows Server 2016. This enables log out of a machine without the server being stopped.

To Register the Server as a Windows Service

Perform the following steps to register the server as a service:

1. Stop the server with `bin/stop-server`. A server cannot be registered while it is running.
2. Register the server as a service. From a Windows command prompt, run `bat/register-windows-service.bat`.
3. After a server is registered, start the server from the Windows Services Control Panel or with the `bat/start-server.bat` command.

   Command-line arguments for the `start-server.bat` and `stop-server.bat` scripts are not supported while the server is registered to run as a Windows service. Using a task to stop the server is also not supported.

To Run Multiple Service Instances

Only one instance of a particular service can run at one time. Services are distinguished by the `wrapper.name` property in the `<server-root>/config/wrapper-product.conf` file. To run additional service instances, change the `wrapper.name` property on each additional instance. Descriptions of the services can also be added or changed in the `wrapper-product.conf` file.

To Deregister and Uninstall Services

While a server is registered as a service, it cannot run as a non-service process or be uninstalled. Use the `bat/deregister-windows-service.bat` file to remove the service from the Windows registry. The server can then be uninstalled with the `uninstall.bat` script.

Log Files for Services

The log files are stored in `<server-root>/logs`, and filenames start with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or due to file size. Only the last three log files are retained. These configurations can be changed in the `<server-root>/config/wrapper.conf` file.

Uninstalling the Server

The Directory Server provides an `uninstall` command-line utility for quick and easy removal of the code base.

To uninstall a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode is a text-based interface that prompts the user for input. You can start the command using the `bin/uninstall` command with the --cli option. The utility prompts you for input if more data is required.
- **Non-Interactive Command-Line Mode**. Non-interactive mode suppresses progress information from being written to standard output during processing, except for fatal errors. This mode is convenient for scripting and is invoked using the `bin/uninstall` command with the --no-prompt option.

Note: For stand-alone installations with a single Directory Server instance, you can also manually remove the Directory Server by stopping the server and recursively deleting the directory and subdirectories. For example:

```
$ rm -rf /ds/PingDirectory
```

To Uninstall the Server in Interactive Mode

Interactive mode uses a text-based, command-line interface to help you remove your instance. If `uninstall` cannot remove all of the Directory Server files, the `uninstall` tool generates a message with a list of the files and directories that must be manually deleted. The `uninstall` command must be run as either the root user or the same user (or role) that installed the Directory Server.

1. From the server root directory, run the `uninstall` command.

```
$ ./uninstall --cli
```

2. Select the components to be removed. If you want to remove all components, press **Enter** to accept the default (remove all). Enter the option to specify the specific components that you want to remove.

```
Do you want to remove all components or select the components to remove?

1) Remove all components
2) Select the components to be removed

q) quit
Enter choice [1]:
```

3. For each type of server component, press **Enter** to remove them or type `no` to keep it.

```
Remove Server Libraries and Administrative Tools? (yes / no) [yes]:
Remove Database Contents? (yes / no) [yes]:
Remove Log Files? (yes / no) [yes]:
Remove Configuration and Schema Files? (yes / no) [yes]:
Remove Backup Files Contained in bak Directory? (yes / no) [yes]:
Remove LDIF Export Files Contained in ldif Directory? (yes / no) [yes]:
```

4. If the Directory Server is part of a replication topology, type `yes` to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a stand-alone server, continue to step 7.
5. Type the Global Administrator ID and password to remove the references to this server in other replicated servers. Then, type or verify the host name or IP address for the server that you are uninstalling.
6. Next, select how you want to trust the server certificate if you have set up SSL or StartTLS. For this example, press **Enter** to accept the default.

```
How do you want to trust the server certificate for the Directory Server
on server.example.com:389?

1) Automatically trust
2) Use a trust store
3) Manually validate

Enter choice [3]:
```

7. If your Directory Server is running, the server is shutdown before continuing the uninstall process. The uninstall processes the removal requests and completes. View the logs for any remaining files. Manually remove any remaining files or directories, if listed.

To Uninstall the Server in Non-Interactive Mode

The `uninstall` utility provides a non-interactive method to enter the command with the --no-prompt option. Another useful argument is the --forceOnError option that continues the uninstall process when an error is encountered. If an option is incorrectly entered or if a required option is omitted and the --forceOnError option is not used, the command will fail and abort.

1. From the server root directory, run `uninstall` tool with the --remove-all option to remove all of the Directory Server's libraries. The --quiet option suppresses output information and is optional. The following command assumes that the Directory Server is stand-alone and not part of a replication topology.

   ```
   $ ./uninstall --cli --remove-all --no-prompt --quiet --forceOnError
   ```
2. If any files or directories remain, manually remove them.

To Uninstall Selected Components in Non-Interactive Mode

From the server root directory, run `uninstall` with the --backup-files option to remove the Directory Server's backup files. Use the --help or -H option to view the other options available to remove specific components.

```
$ ./uninstall --cli --backup-files --no-prompt --quiet --forceOnError
```

Chapter

4

Upgrading the Server

Topics:

Ping Identity issues software release builds periodically with new features, enhancements, and fixes for improved server performance. Administrators can use the Directory Server's update utility to upgrade the current server code version.

This chapter presents some update scenarios and their implications that you should consider when upgrading your server code.

Upgrade Overview and Considerations

The upgrade process involves downloading and unzipping a new version of the Directory Server ZIP file on the server to be updated, and running the `update` utility with the `--serverRoot` or `-R` option value from the new root server pointing to the installation to be upgraded.

Consider the following when upgrading replicating servers:

- Upgrade affects only the server being upgraded. The process does not alter the configuration of other servers.
- The `update` tool will verify that the version of Java that is installed meets the new server requirements. To simplify the process, install the version of Java that is supported by the new server before running the tool.
- To be safe, backup the user data (userRoot) before an upgrade. Restoring from a backup could be necessary if all other servers in the replication topology have been upgraded and a database or encoding change in the new server version prevents the database from being used with the older server version. The `update` and `revert-update` utilities will issue a warning when this is the case.
- Temporarily raise the replication purge delay for all servers in the topology to cover the expected downtime for maintenance. This will result in a temporary increase in disk usage for the replicationChanges database stored in `<server-root>/changelogDb`.
- Replication does not need to be disabled on a server before an upgrade.
- Make sure upgraded servers are working as expected before upgrading the last server in the topology
- Enable new features after all replicating servers are upgraded.

About Updating Servers in a Topology

An update to the current release includes significant changes, and the introduction of a topology registry, which will store information previously stored in the admin backend (server instances, instance and secret keys, server groups, and administrator user accounts). For the admin backend to be migrated, the `update` tool must be provided LDAP authentication options to the peer servers of the server being updated.

The LDAP connection security option requested (either plain, TLS, StartTLS, or SASL) must be configured on every server in the topology. The LDAP credentials must be present on every server in the topology, and must have permissions to read from the admin backend and the config backend of every server in the topology. For example, a root DN user with the `inherit-default-privileges` set to true (such as the `cn=Directory Manager` user) that exists on every server can be used.

After enabling or fixing the configuration of the LDAP connection handler(s) to support the desired connection security mechanism on each server, run the following `dsframework` command on the server being updated so that its admin backend has the most up-to-date information:

```
$ bin/dsframework set-server-properties \
  --serverID serverID \
  --set ldapport:port \
  --set ldapsport:port \
  --set startTLSEnabled:true
```

The `update` tool will verify that the following conditions are satisfied on every server in the topology before allowing the update:

- When the first server is being updated, all other servers in the topology must be online. When updating additional servers, all topology information will be obtained from one of the servers that has already been updated. The `update` tool will connect to the peer servers of the server being updated to obtain the necessary information to populate the topology registry. The provided LDAP credentials must have read permissions to the config and admin backends of the peer servers.
- The instance name is set on every server, and is unique across all servers in the topology. The instance name is a server's identifier in the topology. After all servers in the topology have been updated, each server will be uniquely identified by its instance name. Once set, the name cannot be changed. If needed, the following command can be used to set the instance name of a server prior to the update:

```
$ bin/dsconfig set-global-configuration-prop \
  --set instance-name:uniqueName
```

- The cluster-wide configuration is synchronized on all servers in the topology. Older versions have some topology configuration under `cn=cluster,cn=config` (JSON attribute and field constraints). These items did not support mirrored cluster-wide configuration data. An update should avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. To synchronize the cluster-wide configuration data across all servers in the topology, run the `config-diff` tool on each pair of servers to determine the differences, and use `dsconfig` to update each instance using the `config-diff` output. For example:

```
$ bin/config-diff --sourceHost hostName \
  --sourcePort port \
  --sourceBindDN bindDN \
  --sourceBindPassword password \
  --targetHost hostName \
  --targetPort port \
  --targetBindDN bindDN \
  --targetBindPassword password
```

If any of these conditions are not satisfied, the `update` tool will list all of the errors encountered for each server, and provide instructions on how to fix them.

To Upgrade the Directory Server

Perform an upgrade with the following steps.

1. Download and unzip the new version of the Directory Server in a location outside the existing server's installation. For these steps, assume the existing server installation is in `/prod/PingDirectory` and the new server version is unzipped into `/home/stage/PingDirectory`.
2. Run the `update` tool provided with the new server package to update the existing Directory Server. The update tool may prompt for confirmation on server configuration changes if it detects customization.

```
$ /home/staging/PingDirectory/update --serverRoot /prod/PingDirectory
```

To Upgrade the RPM Package

If the Linux RPM package was used to install the Directory Server, the following should be performed to upgrade the server.

- Assume that the new RPM package, PingDirectory-<new-version>.rpm, is placed in the server root directory. From the server root directory, run the `rpm` command with the --upgrade option.

```
$ rpm --upgrade PingDirectory-<new-version>.rpm
```

The RPM package does not support a revert option once the build is upgraded.

The upgrade history is written to /opt/ping-identity/ds/PingDirectory/PingDirectory/history/<timestamp>/update.log.

Reverting an Update

An installation can be reverted (one level back) using the `revert-update` tool. The `revert-update` tool will place the server’s binaries and configuration back to its prior state. If multiple updates have been performed, the tool can be run multiple times to revert to each prior update sequentially, but only one level at a time.

1. Stop the server and run the `revert-update` command:

```
$ bin/stop-server
$ ./revert-update
```

2. Before starting the server, make sure there were no warnings, such as:

   Warning: Additional steps must be taken once this revert has completed. The current server uses an on disk database format that is not compatible with the server version being reverted to. After performing the revert, compatible databases must be put in place before restarting the server. Different steps must be taken to restore the Local DB Backend, Replication Changelog, and LDAP Changelog databases.

   This type of message indicates that the userRoot, replicationChange and changelog (if enabled) databases need to be restored before starting the server. These databases can be backed up from another server of the same version in the replication topology and restored on the newly reverted server. For example, from another server in the topology, perform the following:

```
$ bin/backup --backendID userRoot --backupDirectory /backups/userRoot
$ bin/backup --backendID replicationChanges --backupDirectory /backups/
replicationChanges
$ bin/backup --backendID changelog --backupDirectory /backups/changelog
```

   Transfer the backups to the reverted server and restore each database with the following commands:

```
$ bin/restore --backupDirectory /transfered/userRoot
$ bin/restore --backupDirectory /transfered/changelog
$ bin/restore --backupDirectory /transfered/replicationChanges
```

3. After all databases have been restored, start the server with the following command:

```
$ bin/start-server
```

Configure SCIM After Upgrade

Modifications in SCIM PATCH are mapped directly to LDAP modifications to use the matching rules configured in the Directory Server, when matching deleted values. Since the SCIM PATCH is now applied by the Directory Server, the Permissive Modify Request Control (1.2.840.113556.1.4.1413) is now required by the SCIM component. This ensures that adding an existing value or deleting a non-existent value in the PATCH request will not generate an error. This affects upgrades from server versions prior to 5.0.0.

To continue using the SCIM component after an upgrade, access controls and configuration must be updated to allow access to the Permissive Modify Request Control. Run the `dsconfig` commands to update these components:

```
$ dsconfig set-access-control-handler-prop \
  --remove 'global-aci:(targetcontrol="1.3.6.1.1.13.2 ||
 1.2.840.113556.1.4.473 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9
 || 1.3.6.1.1.12")(version 3.0;acl "Authenticated access to controls used by
 the SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473
 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9 || 1.3.6.1.1.12 ||
 1.2.840.113556.1.4.1413")(version 3.0;acl "Authenticated access to controls
 used by the SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

Chapter 5

Tuning the Server

Topics:

The PingDirectory Server's installation process automatically determines the optimal Java Virtual Machine (JVM) settings based on calculations of the machine running setup. While the majority of the default configuration and JVM settings are suitable for most deployments, it is not uncommon in high performance environments to make slight adjustments to the Directory Server's JVM settings as well as performance and resource-related configuration changes with the `dsconfig` tool.

This chapter provides guidance for tuning the Directory Server and its tools for both optimum performance with regard to throughput and disk space usage. This chapter presents the following topics:

About Minimizing Disk Access

Most critical to directory server performance is minimizing disk access. Defining a JVM heap size that can contain the entire contents of the database cache in memory is essential to minimizing read operations from disk and achieving optimal performance. It is also important to understand that the database on-disk is comprised of transaction log files, which are only appended to. After an initial database import, the size on-disk will grow by a factor of at least 25% as inactive records accumulate within the transaction logs. Therefore, during normal operation, the on-disk size of the database transaction logs do not represent the memory needed to cache the database.

Another consideration is to minimize the size of the database based on the known characteristics of your data. Minimizing the size of the database not only reduces hard disk requirements but also reduces the memory requirements for the database cache. The Directory Server has the capability to automatically compact common parent DNs, which is an example of optimizing the database size based on known characteristics of the data.

Another consideration is to consider the write load on your server and its affect on the database. While write operations will always require an associated write to disk, an environment that sustains a high load of write operations may consider tuning the background database cleaner to minimize the size of the database on disk.

Memory Allocation and Database Cache

The Directory Server's optimal performance is dependent on the proper allocation of memory to the JVM heap, the number of processor cores in the system, and the correct combination of JVM options for optimized garbage collection. The `setup` tool for the Directory Server automatically assigns the JVM options and determines the memory allocation based on the total amount of memory on the system. However, in most production deployments, additional tuning may be required to meet the performance objectives for your system.

Most often, directory server performance tuning can be accomplished by adjusting a few settings. Tuning these settings, which include both JVM and configuration options, require an understanding of the JVM heap structure as well as the expected database usage. This section describes the basic components of the Directory Server footprint and logic behind the automated tuning of the `setup` tool.

Directory Server Process Memory

The Directory Server is comprised mostly of a JVM heap and a marginal amount of memory allocated by the JVM's execution of native code. While we frequently refer to the JVM Heap as the maximum memory consumed by the Directory Server, the actual process size will be slightly larger than the Xmx value due to accumulation of small chunks of native code that Java requires for things, such as SSL sockets.

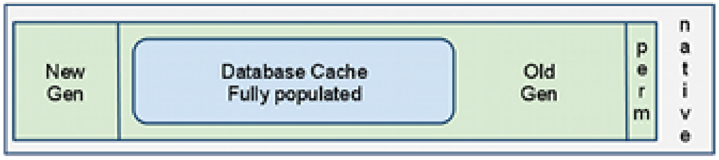


Figure 1: JDK Heap Structure

Within the JVM Heap, the principal memory components are the New and Old Generations. The New Generation is a smaller area of memory where all data is initially allocated and is cleaned of garbage often. Any data that stays "alive" long enough will be promoted to the Old Generation for the longer term. The Old Generation is where the database cache will eventually reside. The Old Generation size is computed from the leftover heap after defining the MaxHeapSize and New Generation sizes; therefore, it is not explicitly stated in the JVM options. A typical set of Generation definitions for the JVM is as follows, where mx and ms values represent the heap size:

```
-Xmx16g -Xms16g -XX:MaxNewSize=2g -XX-NewSize=2g
```

Note: The mx and ms values should always be the same, and the MaxNewSize/NewSize values should be the same. This will help avoid negative changes in performance.

The MaxNewSize/NewSize values should never need to exceed 2g. The `setup` and `dsjavaproperties` tools set MaxNewSize/NewSize values based on the results of extensive performance testing, and should not need to be changed.

A Method for Determining Heap and Database Cache Size

The most straightforward approach to defining the proper memory allocation of the directory server components is to use the Directory Server `setup` command on hardware that represents the target production platform, especially with regard to process and memory, and the largest heap size that the `setup` tool will allow. After running `setup`, any schema and production database settings should be defined in preparation for the database import using the `import-ldif` tool.

Note: At the moment after an `import-ldif`, the database is at its most optimized state on disk with no inactive records. Over time, the on-disk representation of the database will grow as much as 25-50% as inactive records accumulate before being removed by the server's cleaner thread.

After the database is imported, the server should be started and a configuration change made to the backend. In this scenario, set the *prime-method* to "preload" on the userRoot backend configuration. Once the change is made, re-start the Directory Server and watch for a successful preload message at the end of startup. If preloading did not complete, the server should be stopped. The `start-server.java-args` entry in the `config/java.properties` file should be edited to use larger values for `-Xmx` and `-Xms` arguments. Then run the `bin/dsjavaproperties` command and restart the server. If preloading completed successfully, the database cache utilization percentage will be of interest. The `status` command will display something like the following:

```
             --- JE Environment ---
ID    : Cache Full : Cache   : On-Disk : Alert
---------:------------:--------:---------:------
userRoot : 30%     : 1.1 gb : 868.6mb : None
```

Looking at the above output and knowing that the database is fully loaded into cache, the 30% utilization is comfortable cushion for future database growth. In general, it is best to leave at least 10-20% cache headroom available.

During this scenario, it was clear from the `start-server` output that the database primed completely and our interpretation of the `status` output was sound. To see the state of the database cache with more detail, perform an `ldapsearch` on the backend monitor.

In addition to the user configured backends, there may be backends for replication and changelog. The heap is shared among all backends. The amount allocated to each backed is calculated according to the procedure in the next section.

Automatic DB Cache Percentages

The setup process automatically tunes the percentage of the `db-cache-percent` property for the userRoot backend based on the maximum configured JVM heap size. This is only done for the userRoot backend during setup. Other backends created by the user are allocated 10%. The allocation can be changed if needed. When setting up the server, perform the following steps:

- Install the server with necessary memory. The server will autotune the size of the cache.
- Set the autotuned cache size to the limit for the combined cache sizes of all of the backends.
- Divide the server cache based on the expected size of the data in each backend.

Automatic Memory Allocation

If the Memory Tuning feature is enabled during setup, the `setup` algorithm determines the maximum JVM heap size based on the total amount of available system memory. If Memory Tuning is not selected, the server allocates a maximum JVM heap of 384 MB. The Directory Server also allows you to specify the maximum heap size during the setup process. You can enable Memory Tuning during the setup process by selecting the feature during the interactive command-line mode, adding the --jvmTuningParameter option using the `setup` tool in non-interactive command-line mode, or regenerating the java properties file with `bin/dsjavaproperties` and the --jvmTuningParameter options (seen in *JVM Properties for Server and Command-Line Tools* page 75).

If Memory Tuning is selected, the server can allocate the maximum JVM heap depending on the total system memory. The following table displays the automatically allocated maximum JVM heap memory based on available system memory.

Table 2: Allocated Max JVM Memory if Tuning is Enabled

Available Memory	Allocated JVM Memory
16 GB or more using a 64-bit JVM	The maximum JVM heap size will be set to 70% of total system memory. If the maximum JVM heap size is less than or equal to 128GB of memory (which should be the case for systems with up to 160 GB of memory), then the initial heap size will be set to equal the maximum heap size.
6 GB–16 GB using a 64-bit JVM	total system memory - 4 GB
4 GB–6 GB using a 64-bit JVM	2 GB
2 GB–4 GB	512 MB
1 GB–2 GB	384 MB

Automatic Memory Allocation for the Command-Line Tools

At setup, the Directory Server automatically allocates memory to each command-line utility based upon the maximum JVM heap size. The server sets each command-line utility in the `config/java.properties` with -Xmx/Xms values depending on the expected memory needs of the tools. Because some tools can be invoked as a server task while the server is online, there are two definitions of the tool in the `config/java.properties` file: one with .online and one with .offline added to the name. The online invocations of the tools typically require minimal memory as the task is performed within the Directory Server's JVM. The offline invocations of the tools, for example, `import-ldif.offline` and `rebuild-index.offline`, can require the same amount of memory that is needed by the Directory Server.

Beyond the offline tool invocations, some tools, such as `ldap-diff` and `verify-index`, may need more than the minimal memory if large databases are involved. The table below lists the tools that are expected to have more than the minimal memory needs along with the rules for defining the default heap size.

Table 3: Default Memory Allocation to the Command-Line Tools

Command-Line Tools	Allocated JVM Memory
start-server, import-ldif (offline), rebuild-index (offline)	MaxHeapSize
backup (offline), dbtest export-ldif (offline), ldap-diff, restore (offline), scramble-ldif, summarize-access-log, verify-index	If Max System Memory is: Greater than or equal to 16 GB: set Heap to 3 GB Greater than or equal to 8 GB: set Heap to 1 GB Greater than or equal to 4 GB: set Heap to 512 MB Under 4 GB: set Heap to 256 MB

Database Preloading

Key to Directory Server performance is the ability to maintain the database contents in the database cache within the JVM memory. With a properly sized database cache, a priming method of "preload" directs the server to load the database contents into memory at server startup before accepting the first client connection. The time needed to preload the database is proportional to the database size. To avoid priming, the server can be started with the `start-`

`server --skipPrime` command. If the priming method is none or the `--skipPrime` option is specified at startup, the database cache will slowly build as entries are accessed. This could take several days to reach optimal performance.

The "preload" priming method is suitable for nearly all Directory Server deployments. If the size of the database precludes storing the whole database in memory, there are priming alternatives for optimizing server performance. This type of deployment is considered disk-bound since the disk is accessed when processing most operations. See the section Disk-Bound Deployments for more information. The remaining priming options are applicable to these environments.

The Directory Server database `prime-method` property configures how the caches get primed, what gets primed (data, internal nodes, system indexes) and where it gets primed (database cache, filesystem cache, or both). The `prime-method` property is a multi-valued option that enables preloading the internal nodes into the database cache before the server starts, and then primes the values in the background by cursoring across the database. For more details, see the PingDirectory Server Configuration Reference.

The following is a summary of the priming methods:

- **Preload All Data**. Prime the contents of the backend into the database cache.
- **Preload Internal Nodes Only**. Prime only internal database structure information into the database cache, but do not prime any actual data. (This corresponds to the cache-keys-only cache-mode.)
- **Cursor Across Indexes**. Use the `cursor-across-indexes` property to iterate through backend contents. This is similar to (and may be slower than) using the preload mechanism, but it enables priming to happen in the background after the server has started. This is used when shorter start up times are desired, and the slower performance of an uncached database is acceptable until the database is primed.

Configuring Database Preloading

Use the `dsconfig` tool to set the database priming method. If multiple prime methods are used, the order in which they are specified in the configuration is the order in which they will be performed. Changing the preloading option requires re-starting the Directory Server. The following procedure shows how to configure database preloading.

To Configure Database Preloading

1. Set the prime method to "preload" to load the database contents from disk into memory when the server starts up. This eliminates the need for the server to gradually prime the database cache using client traffic, and ensures that the server has optimal performance when it starts to receive client connections.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --set prime-method:preload
```

2. Re-start the Directory Server to apply the changes using `bin/stop-server` and then, `bin/ start-server`.

To Configure Multiple Preloading Methods

1. To achieve the benefits of preloading without delaying server startup, `prime-method` can be set to `preload-internal-nodes-only`, which caches all of the keys within the database but not the values. The database values themselves can be cached in the background once the server has been started with the `cursor-across-indexes` option.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --add prime-method:preload-internal-nodes-only \
  --add prime-method:cursor-across-indexes \
  --set background-prime:true
```

2. Re-start the Directory Server to apply the changes using `bin/stop-server` and then, `bin/start-server`.

To Configure System Index Preloading

1. Some environments have many indexes configured, though only a few are used for performance-sensitive traffic. In this case, server start up time can be reduced by only preloading the necessary indexes into the database at startup.

   ```
   $ bin/dsconfig set-backend-prop --backend-name userRoot \
     --set prime-method:preload \
     --set prime-all-indexes:false \
     --set system-index-to-prime:dn2id \
     --set system-index-to-prime:id2entry
   ```

   ```
   $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
     --index-name mail \
     --set prime-index:true
   ```

   ```
   $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
     --index-name uid \
     --set prime-index:true
   ```

   ```
   $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
     --index-name entryUUID \
     --set prime-index:true
   ```
2. Restart the Directory Server to apply the changes using `bin/stop-server` and then, `bin/start-server`.

Databases on Storage Area Networks, Network-Attached Storage, or running in Virtualized Environments

There are several considerations when using network-based storage or storage abstracted by virtualization that are not issues when databases are stored on local disks. A data durability problem occurs when remote storage or the virtualization environment experiences service interruptions, ranging from connectivity loss to total failure from power loss. Data corruption can occur when the storage layer accepts data for writing that is not made durable before a crash occurs. In these cases, a database property can be set that reduces the likelihood of data loss and data corruption. The database property `database-on-virtualized-or-network-storage` can be set on a per-backend environment basis to request all database writes to be written durably to the underlying storage.

There is a performance penalty when enabling this property, and in most cases, is not recommended except where network storage is unreliable. For network file systems, the benefits of faster recovery and less likelihood of data loss from unplanned events may outweigh the penalty. The exact overhead of enabling `database-on-virtualized-or-network-storage` will depend on the characteristics of the database, the host filesystem, storage array configuration, and network and virtualization input and output parameters. The write overhead penalty may be substantial for SAN environments. Incremental and full backup strategies should be used instead if performance is unacceptable.

To enable `database-on-virtualized-or-network-storage` for each applicable backend, use the following command as an example, which references the configuration for the userRoot backend:

```
$ bin/dsconfig set-global-configuration-prop \
  --set database-on-virtualized-or-network-storage:true
```

This should be set to `false` if the database is on a local disk.

Database Cleaner

Production environments that have a high volume of write operations may require cleaner thread tuning to control the on-disk database size as log files with inactive nodes wait to be cleaned and deleted. The Directory Server stores its Oracle® Berkeley DB Java Edition (JE) database files on-disk in the `db` directory. Each JE database log file is labelled nnnnnnnn.jdb, where nnnnnnnn is an 8-digit hexadecimal number that starts at 00000000 and is increased by 1 for each file written to disk. JE only appends data to the end of each file and does not overwrite any existing data. JE uses one or more cleaner threads that run in the background to compact the number of JE database (db) files.

The cleaner threads begin by scanning the records in each db file, starting with the file that contains the smallest number of active records. Next, the cleaner threads append any active records to the most recent database file. If a record is no longer active due to modifications or deletions, the cleaner threads leave it untouched. After the db file no longer has active records, the cleaner threads can either delete the file or rename the discarded file. Note that because of this approach to cleaning, the database size on-disk can temporarily increase when cleaning is being performed and files are waiting to be removed.

The Local DB Backend configuration object has two properties that control database cleaning: db-cleaner-min-utilization and db-num-cleaner-threads. The db-cleaner-min-utilization property determines, by percentage, when to begin cleaning out inactive records from the database files. By default, the property is set to 75, which indicates that database cleaning ensures that at least 75% of the total log file space is devoted to live data. Note that this property only affects the on-disk representation of the database and not the in-memory database cache—only live data is ever cached in memory.

The db-num-cleaner-threads property determines how many threads are configured for db cleaning. The default single cleaner thread is normally sufficient. However, environments with a high volume of write traffic may need to increase this value to ensure that database cleaning can keep up.

If the number of database files grow beyond your expected guidelines or if the Directory Server is experiencing an increased number of update requests, you can increase the number of cleaner threads using the dsconfig tool (select **Backend** > **select advanced properties** > **db-num-cleaner-threads**).

Compacting Common Parent DNs

The PingDirectory Server compacts entry DNs by tokenizing common parent DNs. Tokenizing the common parent DNs allows you to increase space usage efficiency when encoding entries for storage. The Directory Server automatically defines tokens for base DNs for the backend (for example, dc=example,dc=com). You can also define additional common base DNs that you want to tokenize. For example, use the following configuration to tokenize two branches, ou=people,dc=example,dc=com and ou=customers,dc=example,dc=com:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --add "compact-common-parent-dn:ou=people,dc=example,dc=com" \
  --add "compact-common-parent-dn:ou=customers,dc=example,dc=com"
```

Import Thread Count

For most systems, the default setting of 16 threads is sufficient and provides good import performance. On some systems, increasing the import thread count may lead to improved import performance, while selecting a value that is too large can actually cause import performance to degrade. If minimizing LDIF import time is crucial to your deployment, you must determine the optimal number of import threads for your system, which is dependent on both the underlying system and the dataset being imported.

You can use the dsconfig command to set the number of import threads as follows:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot --set import-thread:24
```

JVM Properties for Server and Command-Line Tools

The Directory Server and tools refer to the config/java.properties file for JVM options that include important memory settings. The java.properties file sets the default Java arguments for the Directory Server and each command-line utility including the default *JAVA_HOME* path.

The `java.properties` is generated at server setup time and defines memory-related JVM settings based on the user-provided value for max heap size if aggressive memory tuning option was selected at setup. Most of the JVM options specified for both server and tools do not need customization after setup. The exception is the -Xmx/Xms options, which specify the maximum and initial JVM heap size. See the section on *Memory Allocation and Database Cache* for advice on tailoring the -Xmx/Xms values.

Other than altering the heap size of the server process (`start-server`) or command-line tools, the most common change required to `java.properties` is when it is desired to update the JVM version. A single edit will apply the new JVM to all server and tool use.

Applying Changes Using dsjavaproperties

To apply the changes to the `config/java.properties` file, edit the file manually, and then run the `bin/dsjavaproperties` utility. The `dsjavaproperties` tool uses the information contained in the `config/java.properties` file to generate a `lib/set-java-home` script (or `lib\set-java-home.bat` on Microsoft Windows systems), which is used by the Directory Server and all of its supporting tools to identify the Java environment and its JVM settings. During the process, `dsjavaproperties` calculates an MD5 digest of the contents of the `config/java.properties` file and stores the digest in the generated `set-java-home` script.

The `dsjavaproperties` utility also performs some minimal validation whenever the property references a valid Java installation by verifying that `$(java-home)/bin/java` exists and is executable.

If you make any changes to the `config/java.properties` file but forget to run `bin/dsjavaproperties`, the Directory Server compares the MD5 digest with the version stored in `set-java-home` and sends a message to standard error if the digests differ:

```
WARNING -- File /ds/PingDirectory/config/java.properties has been edited
 without
running dsjavaproperties to apply the changes
```

To Update the Java Version in the Properties File

To change the version of java that is used by the server and tools, it is necessary to edit the config/java.properties file and apply the change by invoking `bin/dsjavaproperties` with no command line options. Also, the server must be restarted for the change to take affect.

Inside `config/java.properties`, alter the value of `default.java-home` to point to the java correct JRE. Any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration.

```
$ bin/dsjavaproperties
```

To Regenerate the Java Properties File

The `dsjavaproperties` command provides a --initialize option that allows you to regenerate the Java Properties file specifically if you set up the Directory Server using standard memory usage but opt for aggressive memory tuning after setup. Rather than reconfigure the Java Properties file by re-running `setup` or manually editing the `java.properties` file, you can regenerate the properties file for aggressive memory tuning. Any existing file will be renamed with a ".old" suffix.

- Run the `dsjavaproperties` command to regenerate the java properties file for aggressive memory tuning:

  ```
  $ bin/dsjavaproperties --initialize --jvmTuningParameter AGGRESSIVE
  ```

JVM Garbage Collection Using CMS

To ensure reliable server performance with Java, the Directory Server depends on Java's Concurrent Mark and Sweep process (CMS) for background garbage collection. There are several garbage collection options, with CMS being the ideal choice for consistent system availability. The CMS collector runs as one or more background threads, for the most part, within the JVM, freeing up space in JVM Heap from an area called the Old Generation.

One of the criteria used by CMS to determine when to start background garbage collection is a parameter called `CMSInitiatingOccupancyFraction`. This percentage value, which applies to the Old Generation, is a recommendation for the JVM to initiate CMS when data occupancy in Old Generation reaches the threshold.

To understand this CMS property, it is important to know how large the Old Generation is and how much data in the Old Generation is expected to be occupied by the database cache. Ideally, the database cache takes less than 70% of the space available in the Old Generation, and the `CMSInitiatingOccupancyFraction` value of 80 leaves plenty of headroom to prevent the JVM from running out of space in Old Generation due to an inability for CMS to keep up. Because CMS takes processing resources away from the Directory Server, it is not recommended to set the `CMSInitiatingOccupancyFraction` at or below the expected database cache size, which would result in the constant running of CMS in the background. See the section on Memory Footprint and Database Cache for a description of determining Old Generation size.

When the CMS collection process cannot keep pace with memory demands in the Old Generation, the JVM will resort to pausing all application processing to allow a full garbage collection. This event, referred to as a *stop-the-world* pause, does not break existing TCP connections or alter the execution of the Directory Server requests. The goal in tuning CMS is to prevent the occurrence of these pauses. When one does occur, the Directory Server will generate an alert, after the pause, and record the pause time in the error log.

Because determining an ideal `CMSInitiatingOccupancyFraction` can be difficult, the approach we have taken is to warn if the Directory Server detects a garbage collection pause by generating a recommended value for the occupancy threshold based on the current amount of memory being consumed by the backend caches. Unfortunately, it is not possible for an administrator to determine the ideal occupancy threshold value in advance. Therefore, to warn of any impending garbage collection pauses, the Directory Server calculates a recommended value for the `CMSInitiatingOccupancyFraction` property and exposes it in the JVM Memory Usage monitor entry in the following attribute:

```
recommended-cms-initiating-occupancy-fraction-for-current-data-set
```

Also, when you start the server, you will see an administrative alert indicating the current state of the `CMSInitiatingOccupancyFraction` and its recommended value.

```
$ bin/start-server [20/April/2012:10:35:25 -0500] category=CORE
 severity=NOTICE msgID=458886
msg="PingDirectory Server 7.0.1.0 (build 20120418135933Z, R6226) starting up"

... (more output) ...

[20/April/2012:10:35:53 -0500] category=UBID_EXTENSIONS severity=NOTICE
msgID=1880555580 msg="Memory-intensive Directory Server
components are configured to consume 71750382 bytes of memory:
['userRoot local DB backend' currently consumes 26991632 bytes and
can grow to a maximum of 64323584 bytes, 'changelog cn=changelog backend'
currently consumes 232204 bytes and can grow to a maximum of 2426798 bytes,
'Replication Changelog Database' currently consumes 376661 bytes and can
grow to a maximum of 5000000 bytes]. The configured value of
CMSInitiatingOccupancyFraction is 36 which is less than the minimum
recommended value (43) for the server's current configuration. Having
this value too low can cause the Concurrent Mark and Sweep garbage
collector to run too often, which can cause a degradation of throughput
and response time. Consider increasing the CMSInitiatingOccupancyFraction
value to at least the minimum value, preferably setting it to the
recommended value of 55 by editing the config/java.properties file,
running dsjavaproperties, and restarting the Directory Server.
If the server later detects that this setting actually leads to a
performance degradation, a separate warning message will be logged.
If this server has not yet been fully loaded with data, then you
can disregard this message"

[20/April/2012:10:35:53 -0500] category=CORE severity=NOTICE msgID=458887
msg="The Directory Server has started successfully"
```

The Directory Server only makes a recommendation if all of the backends are preloaded and the `CMSInitiatingOccupancyFraction` JVM property is explicitly set, which is done automatically. For example, if you installed the Directory Server and specified that the database be preloaded (or "primed") at startup, then the Directory Server can make a good recommendation for the Directory Server when a pause occurs. If the backend database cache is not full and has not been preloaded, then the recommended value may be an inaccurately low value.

Note: The generated value for the Directory Server property could change over time with each Directory Server build, Java release, or changes in data set. If the current value is fairly close to the recommended value, then there is no need to change the property unless the server experiences a JVM pause.

If the Directory Server experiences a JVM garbage collection pause, you can retrieve the recommended value from the server, reset the Directory Server property, run `dsjavaproperties`, and restart the server.

To Determine the CMSInitiatingOccupanyFraction

1. If you set the Preload Database at startup option during the installation, then skip to step 3. If you are not sure, retrieve the `prime-method` property for the backend as follows:

```
$ bin/dsconfig get-backend-prop --backend-name userRoot \
  --property prime-method
```

2. If the `prime-method` property was not configured, use `bin/dsconfig` to set the property to PRELOAD, and then, restart the Directory Server to preload the database cache.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set prime-method:preload

$ bin/stop-server
$ bin/start-server
```

3. At startup, you will see an administrative message if the current `CMSInitiatingOccupancyFraction` property is below the recommended value. You can get the recommended value from this message and change it in the `config/java.properties` file in step 5.
4. If you were unable to see the recommended `CMSInitiatingOccupancyFraction` property at startup presented in the previous step, first you must pre-tune the value of the `CMSInitiatingOccupancyFraction` property to ensure that all of the data is imported into the server and preloading is enabled in the backend. Next, retrieve the recommended `CMSInitiatingOccupancyFraction` value by issuing the following search. If the `recommended-cms-initiating-occupancy-fraction-for-current-data-set` is not present, then make sure that the server has been restarted since enabling preload for the backend(s).

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectclass=ds-memory-usage-monitor-entry)" \
  cms-initiating-occupancy-fraction \
  recommended-cms-initiating-occupancy-fraction-for-current-data-set
```

```
dn: cn=JVM Memory Usage,cn=monitor
cms-initiating-occupancy-fraction:80
recommended-cms-initiating-occupancy-fraction-for-current-data-set:55
```

5. Open the `config/java.properties` file using a text editor, manually edit the `CMSInitiatingOccupancyFraction` or any other property to its recommended value in the `start-server.java-args` property, and then, save the file when finished. (The following arguments are recommended for a Sun 5440 server. Contact your authorized support provider for specific assistance.):

```
start-server.java-args=-d64 -server -Xmx20g -Xms20g -XX:MaxNewSize=1g
-XX:NewSize=1g -XXParallelGCThreads=16 -XX:+UseConcMarkSweepGC
-XX:+CMSConcurrentMTEnabled -XX:+CMSParallelRemarkEnabled
-XX:+CMSParallelSurvivorRemarkEnabled -XX:ParallelCMSThreads=8
-XX:CMSMaxAbortablePrecleanTime=3600000
-XX:+CMSScavengeBeforeRemark -XX:RefDiscoveryPolicy=1
```

```
-XX:CMSInitiatingOccupancyFraction=55 -XX:+UseParNewGC
-XX:+UseBiasedLocking -XX:+UseLargePages
-XX:+HeapDumpOnOutOfMemoryError
```

The `-XX:ParallelGCThreads` should be limited to 16 (default) or to 8 for smaller systems. Also, the `-XX:ParallelCMSThreads` should be limited to 8.

6. Run the `bin/dsjavaproperties` command to apply the changes.

```
$ bin/dsjavaproperties
```

7. Restart the Directory Server.

Tuning For Disk-Bound Deployments

For best performance, configure the Directory Server to fully cache the DIT in the backend database cache. Directory Server configuration assumes this scenario. For databases too large to fit in memory, other options are available:

- Configure the server for a disk-bound data set (when the database is stored on an SSD, this configuration yields server performance that is comparable to a fully-cached scenario).
- Use uncached attributes and/or entries as described in the following section.
- Use a Directory Proxy Server in an entry-balancing deployment, which allows all data to be cached in a partitioned environment.

To Tune for Disk-Bound Deployments

To configure the server for a disk-bound configuration, follow these steps:

1. When installing the server, choose the "aggressive" option for JVM memory configuration and to preload the data when the server starts.
2. Set the `default-cache-mode` of the `userRoot` backend to `cache-keys-only`.
3. Set operating system `vm.swappiness` to 0 to protect the Directory Server JVM process from an overly aggressive filesystem cache.
4. When the data set is imported with the above settings, verify in the `import-ldif` output that the cached portions of the data set fit comfortably within the database cache.

Uncached Attributes and Entries

Although achieving optimal Directory Server performance requires that the entire data set be fully cached, there may be deployments in which fully caching the data set is not possible due to hardware or financial constraints, or in which acceptable performance can be achieved by only caching a portion of the data. The Directory Server already provides support for controlling caching on a per-database basis (e.g., to cache only certain indexes and/or system databases), but these features may not provide sufficient control over how memory is used, particularly with regard to which entries are included in the cache, and they do not provide any degree of control over caching only a portion of attributes.

To better address the needs of environments that require partial caching, the Directory Server provides two new options: the ability to exclude certain *entries* from the cache, and the ability to exclude certain *attributes* from the cache. The Directory Server uses an `uncached-id2entry` database container, which is similar to the `id2entry` database that maps an entry's unique identifier and its encoded representation. The `uncached-id2entry` database contains either complete and/or partial representations of entries that are intended to receive less memory for caching. For example, if an entry has a particulary large attribute and the system has hardware constraints on memory, then you can configure the system to not cache this particular attribute or entry. This functionality is only available for the local DB backend, which uses the Berkeley DB Java Edition database.

The `uncached-id2entry` database can be included in the set of databases to prime, but if priming is to be performed, it will only include internal nodes and not leaf nodes. For example, the internal nodes of the `uncached-`

id2entry database will be included in the preload if the prime-all-indexes option is set to "true," or if the system-index-to-prime-internal-nodes-only option has a value of "uncached-id2entry".

Backup/Restore. There are no special considerations for backup and restore with regard to uncached entries and attributes. Backup will successfully save your database contents including uncached entries and attributes. Because of the way the server deals with changes to uncached entry and uncached attribute configuration, there is no problem with restoring a backup that was taken with a different uncached entry configuration than is currently in place for the server. Any entries encoded in a manner that is inconsistent with the current uncached entry or uncached attribute configuration will be properly re-encoded whenever they are updated, or whenever the re-encode entries task is invoked.

Replication. Replication does not propagate information about which portions of entries may have been cached or uncached, nor does it require that different replicas have the same uncached attribute or uncached entry configuration.

LDIF Import/Export. When LDIF content is imported into the server, the uncached attribute and uncached entry configuration is used to determine on a per-entry basis whether some or all of the content for that entry should be written into the uncached-id2entry database. The determination is based on the current configuration and is completely independent of and unaware of the configuration that may have been in place when the LDIF data was initially exported. Neither the LDIF import nor export tools provide any options that specifically target only cached or only uncached content, but these tools do provide the ability to include or exclude entries using search filters, or to include or exclude specific attributes.

Server Access Log. Server access log messages may include uncachedDataAcessed=true in the result message for any operation in which it was necessary to access uncached data in the course of processing the associated request. For add, delete, modify, or modify DN result messages, uncachedDataAcessed=true indicates that at least a portion of the new or updated entry was written into the uncached-id2entry database, or that at least a portion of the updated entry was formerly contained in the uncached-id2entry database. For compare result messages, it indicates that at least a portion of the target entry was contained in the uncached-id2entry database and that data from the uncached portion of the entry was required to evaluate the assertion. For search result messages, it indicates that one or more of the entries evaluated as potential matches contained uncached data, and that data from the uncached portion of at least one entry was needed in determining what data should be returned to the client.

Uncached Entry/Attribute Properties. The Directory Server provides three new advanced properties on the Local DB Backend to control the caching mode for the uncached-id2entry database:

- **uncached-id2entry-cache-mode**. Specifies the cache mode that is used when accessing the records in the uncached-id2entry database. If the system has enough memory available to fully cache the internal nodes for this database, then cache-keys-only is recommended, otherwise it is better to select no-caching to minimize the amount of memory required for interacting with the uncached-id2entry database. For more information, see the *PingDirectory Server Configuration Reference*.
- **uncached-attribute-criteria**. Specifies the criteria used to identify attributes that are written into the uncached-id2entry database, rather than the id2entry database. This property is only used for entries in which the associated uncached-entry-criteria does not indicate that the entire entry should be uncached. The property applies to all entry writes, including add, soft delete, modify, and modify DN operations, as well as LDIF import and re-encode processing. Any changes to the property take effect immediately for writes occurring after the change is made. If no value is specified, then all attributes are written into the id2entry database.
- **uncached-entry-criteria**. Specifies the criteria used to identify entries that are written into the uncached-id2entry database, rather than the id2entry database. The property applies to all entry writes, including add, soft delete, modify, and modify DN operations, as well as LDIF import and re-encode processing. Any changes to the property take effect immediately for writes occurring after the change is made. If no value is specified, then all entries are written into the id2entry database.

To Configure Uncached Attributes and Entries

The following procedure assumes that the uncached-id2entry-cache-mode property is set to the default value, cache-keys-only. For more information on the uncached-id2entry cache modes, see the *PingDirectory Server Configuration Reference*.

1. Run `dsconfig` to uncache entries that match the criteria. Here, the filter will uncache all entries that have its location set to "austin" (i.e., `l=austin`).

   ```
   $ bin/dsconfig create-uncached-entry-criteria \
     --criteria-name "Fully Uncached l=austin" --type filter-based \
     --set enabled:true --set "filter:(l=austin)"
   ```
2. Run `dsconfig` to uncache attributes that match the criteria (`attribute-type: jpegPhoto`). The `--type simple` option indicates that the simple uncached attribute criteria be used to specify the attribute-type that should be uncached, which in this example is `jpegPhoto`. For those entries that are fully stored in the `uncached-id2entry` database container, the uncached attribute will be ignored.

   ```
   $ bin/dsconfig create-uncached-attribute-criteria \
     --criteria-name "Uncached jpegPhoto" --type simple \
     --set enabled:true --set attribute-type:jpegPhoto
   ```
3. Set the uncached properties for the `userRoot` backend.

   ```
   $ bin/dsconfig set-backend-prop \
     --backend-name userRoot \
     --set "uncached-entry-criteria:Fully Uncached l=austin" \
     --set "uncached-attribute-criteria:Uncached jpegPhoto"
   ```
4. Run the `re-encode-entries` tool to initiate a task that causes a local DB `userRoot` backend to re-encode all or a specified subset of the entries that it contains. The tool does not alter the entries themselves but provides a useful mechanism for applying significant changes to the way that entries are stored in the backend. The following command initiates a task that re-encodes all fully-cached entries in the `userRoot` backend, rate-limited to no more than 100 entries per second.

   ```
   $ bin/re-encode-entries --hostname directory.example.com --port 389 \
     --bindDN uid=admin,dc=example,dc=com --bindPassword password \
     --backendID userRoot --skipFullyUncachedEntries \
     --skipPartiallyUncachedEntries --ratePerSecond 100
   ```

Chapter 6

Configuring the Server

Topics:

The out-of-the-box, initial configuration settings for the PingDirectory Server provide an excellent starting point for most general-purpose Directory Server applications. However, additional tuning might be necessary to meet the performance, hardware, operating system, and memory requirements for your production environment.

The Directory Server stores its configuration settings in an LDIF file, `config/config.ldif`. Rather than editing the file directly, the Directory Server provides command-line and an Administrative Console to configure the server. The Directory Server also includes tools to create server groups, so that configuration changes can be applied to multiple servers at one time.

This chapter presents the following topics:

Accessing the Directory Server Configuration

The PingDirectory Server configuration can be accessed and modified in the following ways:

- **Using the Administrative Console**. The PingDirectory Server provides an Administrative Console for graphical server management and monitoring. The console provides equivalent functionality as the `dsconfig` command for viewing or editing configurations. All configuration changes using this tool are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands should you need to back out of a configuration.
- **Using the dsconfig Command-Line Tool**. The `dsconfig` tool is a text-based menu-driven interface to the underlying configuration. The tool runs the configuration using three operational modes: interactive command-line mode, non-interactive command-line mode, and batch mode. All configuration changes made using this tool are recorded in `logs/config-audit.log`.

About dsconfig Configuration Tool

The `dsconfig` tool is the text-based management tool used to configure the underlying Directory Server configuration. The tool has three operational modes: interactive mode, non-interactive mode, and batch mode.

The `dsconfig` tool also offers an offline mode using the --offline option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Using dsconfig in Interactive Command-Line Mode

In interactive mode, the `dsconfig` tool offers a filtering mechanism that only displays the most common configuration elements. The user can specify that more expert level objects and configuration properties be shown using the menu system.

Running `dsconfig` in interactive command-line mode provides a user-friendly, menu-driven interface for accessing and configuring the PingDirectory Server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` script without any arguments. You will be prompted for connection and authentication information to the Directory Server, and then a menu will be displayed of the available operation types.

In some cases, a default value will be provided in square brackets. For example, [389] indicates that the default value for that field is port 389. You can press **Enter** to accept the default. To skip the connection and authentication prompts, provide this information using the command-line options of `dsconfig`.

To Configure the Server Using dsconfig Interactive Mode

1. Launch the `dsconfig` tool in interactive command-line mode.

   ```
   $ bin/dsconfig
   ```
2. Next, enter the LDAP connection parameters. Enter the Directory Server host name or IP address, or press **Enter** to accept the default.
3. Enter the number corresponding to the type of LDAP connection (1 for LDAP, 2 for SSL, 3 for StartTLS) that you are using on the Directory Server, or press **Enter** to accept the default (1).
4. Next, type the LDAP listener port number, or accept the default port. The default port is the port number of the server local to the tool.
5. Enter the user bind DN (default, `cn=Directory Manager`) and the bind DN password.
6. On the **Directory Server Configuration Console** main menu, type a number corresponding to the configuration that you want to change. Note that the number can change between releases or within the same release, depending on the options selected (for example, in cases where more expert level objects and and properties are displayed).

In this example, select the number for Backend. Then, set the `db-cache-percent` to 40%. The optimal cache percentage depends on your system performance objectives and must be tuned as determined through analysis. In many cases, the default value chosen by the `setup` utility is sufficient.

7. On the **Backend management** menu, enter the number corresponding to view and edit an existing backend.
8. Select the backend to work with. In this example, using the basic object menu, only one backend that can be viewed in the directory, userRoot. Press **Enter** to accept the default.
9. From the **Local DB Backend** properties menu, type the number corresponding to the `db-cache-percent` property.
10. Enter the option to change the value, and then type the value for the `db-cache-percent` property. In this example, type `40` for "40 %".
11. Review the changes, and then type `f` to apply them.

    Before you apply the change, the `dsconfig` interactive command-line mode provides an option to view the equivalent non-interactive command based on your menu selections. This is useful in building `dsconfig` script files for configuring servers in non-interactive or batch mode. If you want to view the equivalent `dsconfig` non-interactive command, type `d`. For more information, see *Getting the Equivalent dsconfig Non-Interactive Mode Command*.
12. In the **Backend management** menu, type `q` to quit the `dsconfig` tool.

To View dsconfig Advanced Properties

For most configuration settings, some properties are more likely to be modified than others. The `dsconfig` interactive mode provides an option that hides or shows additional advanced properties that administrators might want to configure.

1. Repeat steps 1–9 in the previous section using `dsconfig` in Interactive Command-Line Mode.
2. From the **Local DB Backend properties** menu, type `a` to display the advanced properties, which toggles any hidden properties.

Using dsconfig Interactive Mode: Viewing Object Menus

Because some configuration objects are more likely to be modified than others, the PingDirectory Server provides four different object menus that hide or expose configuration objects to the user. The purpose of object levels is to simply present only those properties that an administrator will likely use. The Object type is a convenience feature designed to unclutter menu readability.

The following object menus are available:

- **Basic**. Only includes the components that are expected to be configured most frequently.
- **Standard**. Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.
- **Advanced**. Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be potentially harmful if configured incorrectly.
- **Expert**. Includes all components in the Basic, Standard, and Advanced menus plus other components that should almost never require configuration or that could seriously impact the functionality of the server if not properly configured.

To Change the dsconfig Object Menu

1. Repeat steps 1–6 in the section using `dsconfig` in To Install the Directory Server in Interactive Mode.
2. On the **PingDirectory Server configuration** main menu, type **o** (letter “o”) to change the object level. By default, Basic objects are displayed.
3. Enter a number corresponding to a object level of your choice: `1` for Basic, `2` for Standard, `3` for Advanced, `4` for Expert.
4. View the menu at the new object level. Additional configuration options for the Directory Server components are displayed.

Using dsconfig Interactive: Viewing Administrative Alerts

The `dsconfig` tool and the Administrative Console provide a useful feature that displays notifications for certain operations that require further administrator action to complete the process. If you change a certain backend configuration property, the admin action will appear in two places during a `dsconfig` interactive session: when configuring the property and before you apply the change. For example, if you change the `db-directory` property on the userRoot backend (that is, specify the path to the filesystem path that holds the Oracle Berkeley DB Java Edition backend files), you will see an admin action reminder during one of the steps (shown below).

The admin action alert will also appear as a final confirmation step. The alert allows you to continue and apply the change or back out of the configuration if the resulting action cannot be conducted at the present time. For example, after you type "`f`" to apply the `db-directory` property change, the admin alert message appears:

```
Enter choice [b]: f
One or more configuration property changes require administrative action or
confirmation/notification.

Those properties include:

* db-directory: Modification requires that the Directory Server be stopped,
  the database directory manually relocated, and then the Directory Server
  restarted. While the Directory Server is stopped, the directory and files
  pertaining to this backend in the old database directory must be manually
  moved or copied to the new location.

Continue? Choose 'no' to return to the previous step (yes / no) [yes]:
```

Currently, only a small set of properties display an admin action alert appear in `dsconfig` interactive mode and the Administrative Console. For more information on the properties, see the *PingDirectory Server Configuration Reference*.

Using dsconfig in Non-Interactive Mode

The `dsconfig` non-interactive command-line mode provides a simple way to make arbitrary changes to the Directory Server by invoking it from the command line. To use administrative scripts to automate configuration changes, run the `dsconfig` command in non-interactive mode, which is convenient scripting applications. Note, however, that if you plan to make changes to multiple configuration objects at the same time, then the batch mode might be more appropriate.

You can use the `dsconfig` tool to update a single configuration object using command-line arguments to provide all of the necessary information. The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The --no-prompt argument indicates that you want to use non-interactive mode. The {sub-command} is used to indicate which general action to perform. The {globalArgs} argument provides a set of arguments that specify how to connect and authenticate to the Directory Server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on your setup. For example, using standard LDAP connections, you can invoke the `dsconfig` tool as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

If your system uses SASL GSSAPI (Kerberos), you can invoke `dsconfig` as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The {subcommandArgs} argument contains a set of arguments specific to the particular subcommand that you wish to invoke. To always display the advanced properties, use the --advanced command-line option.

Note: Global arguments can appear anywhere on the command line (including before the subcommand, and after or intermingled with subcommand-specific arguments). The subcommand-specific arguments can appear anywhere after the subcommand.

To Configure the Server Using dsconfig Non-Interactive Mode

- Use the `dsconfig` command in non-interactive mode to change the amount of memory used for caching database contents and to specify common parent DNs that should be compacted in the underlying database.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --set db-cache-percent:40 \
  --add "compact-common-parent-dn:ou=accts,dc=example,dc=com" \
  --add "compact-common-parent-dn:ou=subs,dc=example,dc=com"
```

To View a List of dsconfig Properties

1. Use the `dsconfig` command with the list-properties option to view the list of all `dsconfig` properties. Remember to add the LDAP connection parameters.

```
$ bin/dsconfig list-properties
```

2. Use the `dsconfig` command with the list-properties option and the --complexity <menu level> to view objects at and below the menu object level. You can also add the --includeDescription argument that includes a synopsis and description of each property in the output. Remember to add the LDAP connection parameters.

```
$ bin/dsconfig list-properties --complexity advanced --includeDescription
```

3. If the server is offline, you can run the command with the --offline option. You do not need to enter the LDAP connection parameters.

```
 $ bin/dsconfig list-properties --offline --complexity advanced --
includeDescription
```

You can also view the `<server-root>/docs/config-properties.txt` that contains the property information provided with the server.

Getting the Equivalent dsconfig Non-Interactive Mode Command

While the `dsconfig` non-interactive command-line mode is convenient for scripting and automating processes, obtaining the correct arguments and properties for each configuration change can be quite time consuming.

To facilitate easy and quick configuration, you can use an option to display the equivalent non-interactive command using `dsconfig` interactive mode. The command displays the equivalent `dsconfig` command to recreate the configuration in a scripted configuration or to more quickly enter any pending changes on the command line for another server instance.

Note: There are two other ways to get the equivalent `dsconfig` command. One way is by looking at the `logs/config-audit.log`. It might be more convenient to set the Directory Server up the way you want and then get the `dsconfig` arguments from the log. Another way is by configuring an option using the Administrative Console. The console shows the equivalent `dsconfig` command prior to applying the change.

To Get the Equivalent dsconfig Non-Interactive Mode Command

1. Using `dsconfig` in interactive mode, make changes to a configuration but do not apply the changes (that is, do not enter "f").
2. Enter `d` to view the equivalent non-interactive command.
3. View the equivalent command (seen below), and then press **Enter** to continue. For example, based on an example in the previous section, changes made to the `db-cache-percent` returns the following:

   ```
   Command line to apply pending changes to this Local DB Backend:
   dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:40
   ```

   The command does not contain the LDAP connection parameters required for the tool to connect to the host since it is presumed that the command would be used to connect to a different remote host.

Using dsconfig Batch Mode

The PingDirectory Server provides a `dsconfig` batching mechanism that reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The --no-prompt option is required with `dsconfig` in batch mode.

If a `dsconfig` command has a missing or incorrect argument, the command will fail and abort the batch process without applying any changes to the Directory Server. The `dsconfig` command supports a `--batch-continue-on-error` option which instructs `dsconfig` to apply all changes and skip any errors.

You can view the `logs/config-audit.log` file to review the configuration changes made to the Directory Server and use them in the batch file. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The Directory Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to PingDirectory Server machines.

To Configure the Directory Server in dsconfig Batch Mode

1. Create a text file that lists each `dsconfig` command with the complete set of properties that you want to apply to the Directory Server. The items in this file should be in the same format as those accepted by the `dsconfig` command. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

   ```
   # This dsconfig operation creates the exAccountNumber global attribute
    index.
   dsconfig create-global-attribute-index
   --processor-name ou_people_dc_example_dc_com-eb-req-processor
   --index-name exAccountNumber --set prime-index:true

   # Here we create the entry-count placement algorithm with the
   # default behavior of adding entries to the smallest backend
   # dataset first.

   dsconfig create-placement-algorithm
   --processor-name ou_people_dc_example_dc_com-eb-req-processor
   --algorithm-name example_com_entry_count
   --type entry-counter
   --set enabled:true
   --set "poll-interval:1 m"

   # Note that once the entry-count placement algorithm is created
   ```

```
# and enabled, we can delete the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then delete the
# original round-robin algorithm created during the setup
# procedure.

dsconfig delete-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin
```

2. Use `dsconfig` with the --batch-file option to read and execute the commands.

About Recurring Tasks and Task Chains

For regular maintenance items that need to be done on the PingDirectory Server, recurring tasks and task chains can be created with the `dsconfig create-recurring-task` command. These tasks can be created to perform regular backups, LDIF exports, or other static operations. Because this process is owned by the server, tasks do not require special priviledges or credentials, and they can be run when the server is offline. Tasks are created and then added to a recurring task chain for scheduling. The task chain insures that that invocations of a task or set of tasks run in a specified order and cannot overlap.

A recurring task includes:

- The task-specific object classes to include in the task entry.
- The task-specific attributes to include in the task entry, if any.
- Whether to alert on task start, success, and/or failure.
- Any addresses to email on task start, success, and/or failure.
- Whether to cancel an instance of the task if it is dependent upon another task, and that task does not complete successfully.

Once a task is created, one or more tasks can be added to and scheduled with a task chain with the `dsconfig create-recurring-task-chain` command. A recurring task chain includes:

- An ordered list of the tasks to invoke.
- The months, days, times, and time zones in which the task can be scheduled to start.
- The behavior to exhibit if any of the tasks are interrupted by a server shutdown.
- The behavior to exhibit if the server is offline when the start time occurs.

To Create a Recurring Task and Task Chain

Use `dsconfig` to create one or more tasks and then add them to a task chain for scheduling.

1. Create a task. The following creates a backup task.

```
$ bin/dsconfig create-recurring-task \
  --task-name backup-1 \
  --type backup \
  --set 'email-on-failure:admin2@company.com' \
  --set 'email-on-failure:admin@company.com' \
  --set compress:true \
  --set encrypt:true \
  --set "retain-previous-full-backup-age:4 w" \
  --set retain-previous-full-backup-count:10
```

2. Create a task chain to schedule and run recurring tasks.

```
$ bin/dsconfig create-recurring-task-chain \
  --chain-name "backup chain" \
  --set recurring-task:backup-1 \
  --set scheduled-date-selection-type:selected-days-of-the-month \
  --set scheduled-day-of-the-month:last-day-of-the-month \
  --set scheduled-time-of-day:02:00
```

Topology Configuration

Topology configuration enables grouping servers and mirroring configuration changes automatically. It uses a master/slave architecture for mirroring shared data across the topology. All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group. Servers can be added to an existing topology at installation.

Note: To remove a server from the topology, it must be uninstalled with the uninstall tool.

Topology Master Requirements and Selection

A topology master server receives any configuration change from other servers in the topology, verifies the change, then makes the change available to all connected servers. The master always sends a digest of its subtree contents on each update. If the node has a different digest than the master, it knows it's not synchronized. The servers will pull the entire subtree from the master if they detect that they are not synchronized. A server may detect it is not synchronized with the master under the following conditions:

- At the end of its periodic polling interval, if a server's subtree digest differs from that of its master, then it knows it's not synchronized.
- If one or more servers have been added to or removed from the topology, the servers will not be synchronized.

The master of the topology is selected by prioritizing servers by minimum supported product version, most available, newest server version, earliest start time, and startup UUID (a smaller UUID is preferred).

After determining a master, the topology data is reviewed from all available servers (every five seconds by default) to determine if any new information makes a server better suited to being the master. If a new server can be the master, it will communicate that to the other servers, if no other server has advertised that it should be the master. This ensures that all servers accept the same master at approximately the same time (within a few milliseconds of each other). If there is no better master, the initial master maintains the role.

After the best master has been selected for the given interval, the following conditions are confirmed:

- A majority of servers is reachable from that master. (The master server itself is considered while determining this majority.)
- There is only a single master in the entire topology.

If either of these conditions is not met, the topology is without a master and the peer polling frequency is reduced to 100 milliseconds to find a new master as quickly as possible. If there is no master in the topology for more than one minute, a `mirrored-subtree-manager-no-master- found` alarm is raised. If one of the servers in the topology is forced as master with the `force-as-master-for-mirrored-data` option in the Global Configuration object, a `mirrored-subtree-manager-forced-as-master-warning` warning alarm is raised. If multiple servers have been forced as masters, then a `mirrored-subtree-manager-forced-as-master-error` critical alarm will be raised.

Topology Components

When a server is installed, it can be added to an existing topology, which will clone the server's configuration. Topology settings are designed to operate without additional configuration. If required, some settings can be adjusted to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the Global Configuration and in the Config File Handler Backend. Though they are topology settings, they are unique to each server and are not mirrored. Settings must be kept the same on all servers.

The Global Configuration object contains a single topology setting, force-as-master-formirrored- data. This should be set to true on only one of the servers in the topology, and is used only if a situation occurs where the topology cannot determine a master because a majority of servers is not available. A server with this setting enabled will be assigned the role of master, if no suitable master can be determined.

The Config File Handler Backend defines three topology (`mirrored-subtree`) settings:

- `mirrored-subtree-peer-polling-interval` – Specifies the frequency at which the server polls its topology peers to determine if there are any changes that may warrant a new master selection. A lower value will ensure a faster failover, but it will also cause more traffic among the peers. The default value is five seconds. If no suitable master is found, the polling frequency is adjusted to 100 milliseconds until a new master is selected.
- `mirrored-subtree-entry-update-timeout` – Specifies the maximum length of time to wait for an update operation (add, delete, modify or modify-dn) on an entry to be applied by the master on all of the servers in the topology. The default is 10 seconds. In reality, updates can take up to twice as much time as this timeout value if master selection is in progress at the time the update operation was received.
- `mirrored-subtree-search-timeout` – Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

Topology meta-data is stored under the `cn=topology,cn=config` subtree and cluster data is stored under the `cn=cluster,cn=config` subtree. The only setting that can be changed is the cluster name.

Monitor Data for the Topology

Each server has a monitor that exposes that server's view of the topology in its monitor backend, so that peer servers can periodically read this information to determine if there are changes in the topology. Topology data includes the following:

- The server ID of the current master, if the master is not known.
- The instance name of the current master, or if a master is not set, a description stating why a master is not set.
- A flag indicating if this server thinks that it should be the master.
- A flag indicating if this server is the current master.
- A flag indicating if this server was forced as master.
- The total number of configured peers in the topology group.
- The peers connected to this server.
- The current availability of this server.
- A flag indicating whether or not this server is not synchronized with its master, or another node in the topology if the master is unknown.
- The amount of time in milliseconds where multiple masters were detected by this server.
- The amount of time in milliseconds where no suitable server is found to act as master.
- A SHA-256 digest encoded as a base-64 string for the current subtree contents.

The following metrics are included if this server has processed any operations as master:

- The number of operations processed by this server as master.
- The number of operations processed by this server as master that were successful.
- The number of operations processed by this server as master that failed to validate.
- The number of operations processed by this server as master that failed to apply.
- The average amount of time taken (in milliseconds) by this server to process operations as the master.
- The maximum amount of time taken (in milliseconds) by this server to process an operation as the master.

Updating the Server Instance Listener Certificate

To change the SSL certificate for the server, update the keystore and truststore files with the new certificate. The certificate file must have the new certificate in PEM-encoded format, such as:

```
-----BEGIN CERTIFICATE-----

 MIIDKTCCAhGgAwIBAgIEacgGrDANBgkqhkiG9w0BAQsFADBFMR4wHAYDVQQKExVVbmJvdW5kSUQgQ2VydGlmaWN
 GUxIzAhBgNVBAMTGnZtLW1lZGl1bS03My51bmJvdW5kaWQubGFiMB4XDTE1MTAxMjE1MzU0OFoXDTM1MTAwNzE1

 U0OFowRTEeMBwGA1UEChMVVW5ib3VuZElEIENlcnRpZmljYXRlMSMwIQYDVQQDExp2bS1tZWRpdW0tNzMudW5ib

 uZGlkLmxhYjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKN4tAN3o9Yw6Cr9hivwVDxJqF6+aEi9Ir
            GFYLSrggRNXsiAOfWkSMWdIC5vyF5OJ9DlIgvHL4OuqP/
YNEGzKDkgr6MwtUeVSK14+dCixygJGC0nY7k+f0WSCjt
            IHzrmc4WWdrZXmgb
+qv9LupS30JG0FXtcbGkYpjaKXIEqMg4ekz3B5cAvE0SQUFyXEdN4rWOn96nVFkb2CstbiPzA

 gne2tu7paJ6SGFOW0UF7v018XY1m2WHBIoD0WC8nOVLTG9zFUavaOxtlt1TlhClkI4HRMNg8n2EtSTdQRizKuw9
            TXJBb6Kfvnp/
nI73VHRyt47wUVueehEDfLtDP8pMCAwEAAaMhMB8wHQYDVR0OBBYEFMrwjWxl2K+yd9+Y65oKn0g5
            jITgMA0GCSqGSIb3DQEBCwUAA4IBAQBpsBYodblUGew+HewqtO2i8Wt+vAbt31zM5/
kRvo6/+iPEASTvZdCzIBcgl

 etxKGKeCQ0GPeHr42+erakiwmGDlUTYrU3LU5pTGTDLuR2IllTT5xlEhCWJGWipW4q3Pl3cX/9m2ffY/
JLYDfTJao

 JvnXrh7Sg719skkHjWZQgOHXlkPLx5TxFGhAovE1D4qLVRWGohdpWDrIgFh0DVfoyAn1Ws9ICCXdRayajFI4Lc6
            m6SA5+25Y9nno8BhVPf4q5OW6+UDc8MsLbBsxpwvR6RJ5cv3ypfOriTehJsG
+9ZDo7YeqVsTVGwAlW3PiSd9bYP/8
            yu9Cy+0MfcWcSeAE
            -----END CERTIFICATE-----
```

If clients that already have a secure connection established with this server need to be maintained, information about both certificates can reside in the same file (each with their own begin and end headers and footers).

After the keystore and truststore files are updated, run the following dsconfig command to update the server's certificate in the topology registry:

```
$ bin/dsconfig set-server-instance-listener-prop \
  --instance-name server-instance-name \
  --listener-name ldap-listener-mirrored-config \
  --set listener-certificate path-to-new-certificate-file
```

The `listener-certificate` in the topology registry is like a trust store. The public certificates that it has are automatically trusted by the local server. When the local server attempts a secure LDAP connection to a peer, and the peer presents it with its certificate, the local server will check the `listener-certificate` property for that server in the topology registry. If the property contains the peer server's certificate, the local server will trust the peer.

Remove the Self-signed Certificate

The server is installed with a self-signed certificate and key (`ads-certificate`), which are used for internal purposes such as replication authentication, inter-server authentication in the topology registry, reversible password encryption, and encrypted backup or LDIF export. The ads-certificate lives in the keystore file called `ads-truststore` under the server's `/config` directory. If your deployment requires removing the self-signed certificate, it can be replaced.

The certificate is stored in the topology registry, which enables replacing it on one server and having it mirrored to all other servers in the topology. Any change is automatically mirrored on other servers in the topology. It is stored in human-readable PEM-encoded format and can be updated with dsconfig. The following general steps are required to replace the self-signed certificate:

1. Prepare a new keystore with the replacement key-pair.
2. Update the server configuration to use the new certificate by adding it to the server's list of certificates in the topology registry so that it is trusted by other servers.
3. Update the server's `ads-truststore` file to use the new key-pair.
4. Retire the old certificate by removing it from the topology registry.

Note: Replacing the entire key-pair instead of just the certificate associated with the original private key can make existing backups and LDIF exports invalid. This should be performed immediately after setup or before the key-pair is used. After the first time, only the certificate associated with the private key should have to be changed, for example, to extend its validity period or replace it with a certificate signed by a different CA.

Prepare a New Keystore with the Replacement Key-pair

The self-signed certificate can be replaced with an existing key-pair, or the certificate associated with the original key-pair can be used.

To Use an Existing Key-pair

If a private key and certificate(s) in PEM-encoded format already exist, both the original private key and self-signed certificate can be replaced in ads-truststore with the `manage-certificates` tool.

- The following command imports the keystore file, ads-truststore.new.

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file existing.key \
  --certificate-file existing.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

The certificates listed using the `--certificate-file` options must be ordered so that each subsequent certificate is the issuer for the previous one. So the server certificate comes first, the intermediate certificates next (if any), and the root CA certificate last.

To Use the Certificate Associated with the Original Key-pair

The certificate associated with the original server-generated private key can be replaced with the following commands:

1. Create a CSR for the `ads-certificate`:

```
$ bin/manage-certificates generate-certificate-signing-request \
  --keystore ads-truststore \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --use-existing-key-pair \
  --subject-dn "CN=ldap.example.com,O=Example Corporation,C=US" \
  --output-file ads.csr
```

2. Submit `ads.csr` to a CA for signing.
3. Export the server's private key into `ads.key`:

```
$ bin/manage-certificates export-private-key \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --output-file ads.key
```

4. Import the certificates obtained from the CA (the CA-signed server certificate, any intermediate certificates, and root CA certificate) into `ads-truststore.new`:

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file ads.key \
  --certificate-file new-ads.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

To Update the Server Configuration to Use the New Certificate

To update the server to use the desired key-pair, the `inter-server-certificate` property for the server instance must first be updated in the topology registry. The old and the new certificates may appear within their own begin and end headers in the `inter-server-certificate` property to support transitioning from the old certificate to the new one.

1. Export the server's old ads-certificate into `old-ads.crt`::

```
$ bin/manage-certificates export-certificate \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --export-certificate-chain \
  --output-file old-ads.crt
```

2. Concatenate the old, new certificate, and issuer certificates into one file. On Windows, an editor like notepad can be used. On Unix platforms, use the following command:

```
$ cat old-ads.crt new-ads.crt intermediate.crt root-ca.crt > chain.crt
```

3. Update the `inter-server-certificate` property for the server instance in the topology registry using `dsconfig`:

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name instance-name \
  --set “inter-server-certificate<chain.crt”
```

To Update the ads-truststore File to Use the New Key-pair

The server will still use the old `ads-certificate`. When the new `ads-certificate` needs to go into effect, the old `ads-truststore` file must be replaced with `ads-truststore.new` in the server's `config` directory.

- Move the file.

```
$ mv ads-truststore.new ads-truststore
```

To Retire the Old Certificate

The old certificate is retired by removing it from the topology registry when it has expired. All existing encrypted backups and LDIF exports are not affected because the public key in the old and new server certificates are the same, and the private key will be able to decrypt them.

- Perform the following commands:

```
$ cat new-ads.crt intermediate.crt root-ca.crt<chain.crt
```

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name instance-name \
  --set “inter-server-certificate<chain.crt”
```

Using the Configuration API

PingDirectory Server provides a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API is consistent with the System for Cross-domain Identity

Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers should allow the application/json content type.

The server includes a servlet extension that provides read and write access to the server's configuration over HTTP. The extension is enabled by default for new installations, and can be enabled for existing deployments by simply adding the extension to one of the server's HTTP Connection Handlers, as follows:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add http-servlet-extension:Configuration
```

The API is made available on the HTTPS Connection handler's host:port in the `/config` context. Due to the potentially sensitive nature of the server's configuration, the HTTPS Connection Handler should be used for hosting the Configuration extension.

Authentication and Authorization with the Configuration API

Clients must use HTTP Basic authentication to authenticate to the Configuration API. If the username value is not a DN, then it will be resolved to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper, or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. For example:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Configuration \
  --set "identity-mapper:Alternative Identity Mapper”
```

To access configuration information, users must have the appropriate privileges:

- To access the `cn=config` backend, users must have the `bypass-acl` privilege or be allowed access to the configuration using an ACI.
- To read configuration information, users must have the `config-read` privilege.
- To update the configuration, users must have the `config-write` privilege.

Relationship Between the Configuration API and the dsconfig Tool

The Configuration API is designed to mirror the `dsconfig` tool, using the same names for properties and object types. Property names are presented as hyphen case in `dsconfig` and as camel-case attributes in the API. In API requests that specify property names, case is not important. Therefore, `baseDN` is the same as `baseDn`. Object types are represented in hyphen case. API paths mirror what is in `dsconfig`. For example, the `dsconfig list-connection-handlers` command is analogous to the API's `/config/connection-handlers` path. Object types that appear in the schema URNs adhere to a `type:subtype` syntax. For example, a Local DB Backend's schema URN is `urn:unboundid:schemas:configuration:2.0:backend:local-db`. Like the `dsconfig` tool, all configuration updates made through the API are recorded in `logs/config-audit.log`.

The API includes the filter, sort, and pagination query parameters described by the SCIM specification. Specific attributes may be requested using the attributes query parameter, whose value must be a comma-delimited list of properties to be returned, for example `attributes=baseDN,description`. Likewise, attributes may be excluded from responses by specifying the `excludedAttributes` parameter.

Operations supported by the API are those typically found in REST APIs:

HTTP Method	Description	Related dsconfig Example
GET	Lists the properties of an object when used with a path representing an object, such as `/config/global-configuration` or `/config/backends/userRoot`. Can also list objects when used with a path representing a parent relation, such as `/config/backends`.	`get-backend-prop`, `list-backends`, `get-global-configuration-prop`

HTTP Method	Description	Related dsconfig Example
POST	Creates a new instance of an object when used with a relation parent path, such as `/config/backends`.	`create-backend`
PUT	Replaces the existing properties of an object. A PUT operation is similar to a PATCH operation, except that the PATCH identifies the difference between an existing target object and a supplied source object. Only those properties in the source object are modified in the target object. The target object is specified using a path, such as `/config/backends/userRoot`.	`set-backend-prop`, `set-global-configuration-prop`
PATCH	Updates the properties of an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`set-backend-prop`, `set-global-configuration-prop`
DELETE	Deletes an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`delete-backend`

The OPTIONS method can also be used to determine the operations permitted for a particular path.

Object names, such as `userRoot` in the Description column, must be URL-encoded for use in the path segment of a URL. For example, `%20` must be used in place of spaces, and `%25` is used in place of the percent (%) character. The URL for accessing the HTTP Connection Handler object is:

```
/config/connection-handlers/http%20connection%20handler
```

GET Example

The following is a sample GET request for information about the `userRoot` backend:

```
GET /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
```

The response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
   "id": "userRoot",
   "meta": {
     "resourceType": "Local DB Backend",
     "location": "http://localhost:5033/config/backends/userRoot"
   },
   "backendID": "userRoot2",
   "backgroundPrime": "false",
   "backupFilePermissions": "700",
   "baseDN": [
     "dc=example2,dc=com"
   ],
   "checkpointOnCloseCount": "2",
   "cleanerThreadWaitTime": "120000",
   "compressEntries": "false",
   "continuePrimeAfterCacheFull": "false",
   "dbBackgroundSyncInterval": "1 s",
   "dbCachePercent": "10",
   "dbCacheSize": "0 b",
   "dbCheckpointerBytesInterval": "20 mb",
   "dbCheckpointerHighPriority": "false",
   "dbCheckpointerWakeupInterval": "1 m",
   "dbCleanOnExplicitGC": "false",
   "dbCleanerMinUtilization": "75",
   "dbCompactKeyPrefixes": "true",
   "dbDirectory": "db",
   "dbDirectoryPermissions": "700",
   "dbEvictorCriticalPercentage": "0",
   "dbEvictorLruOnly": "false",
   "dbEvictorNodesPerScan": "10",
   "dbFileCacheSize": "1000",
   "dbImportCachePercent": "60",
   "dbLogFileMax": "50 mb",
   "dbLoggingFileHandlerOn": "true",
   "dbLoggingLevel": "CONFIG",
   "dbNumCleanerThreads": "0",
   "dbNumLockTables": "0",
   "dbRunCleaner": "true",
   "dbTxnNoSync": "false",
   "dbTxnWriteNoSync": "true",
   "dbUseThreadLocalHandles": "true",
   "deadlockRetryLimit": "10",
   "defaultCacheMode": "cache-keys-and-values",
   "defaultTxnMaxLockTimeout": "10 s",
   "defaultTxnMinLockTimeout": "10 s",
   "enabled": "false",
   "explodedIndexEntryThreshold": "4000",
   "exportThreadCount": "0",
   "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
   "externalTxnDefaultMaxLockTimeout": "100 ms",
   "externalTxnDefaultMinLockTimeout": "100 ms",
   "externalTxnDefaultRetryAttempts": "2",
   "hashEntries": "false",
   "id2childrenIndexEntryLimit": "66",
   "importTempDirectory": "import-tmp",
   "importThreadCount": "16",
   "indexEntryLimit": "4000",
   "isPrivateBackend": "false",
   "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
   "jeProperty": [
     "je.cleaner.adjustUtilization=false",
     "je.nodeMaxEntries=32"
   ],
   "numRecentChanges": "50000",
   "offlineProcessDatabaseOpenTimeout": "1 h",
   "primeAllIndexes": "true",
   "primeMethod": [
     "none"
   ],
   "primeThreadCount": "2",
   "primeTimeLimit": "0 ms",
   "processFiltersWithUndefinedAttributeTypes": "false",
   "returnUnavailableForUntrustedIndex": "true",
   "returnUnavailableWhenDisabled": "true",
   "setDegradedAlertForUntrustedIndex": "true",
   "setDegradedAlertWhenDisabled": "true",
   "subtreeDeleteBatchSize": "5000",
   "subtreeDeleteSizeLimit": "5000",
   "uncachedId2entryCacheMode": "cache-keys-only",
   "writabilityMode": "enabled"
}
```

GET List Example

The following is a sample GET request for all local backends:

```
GET /config/backends/
Host: example.com:5033
Accept: application/scim+json
```

The response (which has been shortened):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=topology,cn=config"
      ],
      "enabled": "true",
      "isPrivateBackend": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.LDIFBackend",
      "ldifFile": "config/admin-backend.ldif",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "false",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
      ],
      "id": "ads-truststore",
      "meta": {
        "resourceType": "Trust Store Backend",
        "location": "http://localhost:5033/config/backends/ads-
truststore"
      },
      "backendID": "ads-truststore",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=ads-truststore"
      ],
      "enabled": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.TrustStoreBackend",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "true",
      "trustStoreFile": "config/server.keystore",
      "trustStorePin": "********",
      "trustStoreType": "JKS",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:alarm"
      ],
      "id": "alarms",
      "meta": {
        "resourceType": "Alarm Backend",
        "location": "http://localhost:5033/config/backends/alarms"
      },
  ...
```

PATCH Example

Configuration can be modified using the HTTP PATCH method. The PATCH request body is a JSON object formatted according to the SCIM patch request. The Configuration API, supports a subset of possible values for the path attribute, used to indicate the configuration attribute to modify.

The configuration object's attributes can be modified in the following ways. These operations are analogous to the `dsconfig modify-[object]` options.

- An operation to set the single-valued `description` attribute to a new value:

```
{
  "op" : "replace",
  "path" : "description",
  "value" : "A new backend."
}
```

is analogous to:

```
$ dsconfig set-backend-prop
  --backend-name userRoot \
  --set "description:A new backend"
```

- An operation to add a new value to the multi-valued `jeProperty` attribute:

```
{
  "op" : "add",
  "path" : "jeProperty",
  "value" : "je.env.backgroundReadLimit=0"
}
```

is analogous to:

```
$ dsconfig  set-backend-prop --backend-name userRoot \
  --add je-property:je.env.backgroundReadLimit=0
```

- An operation to remove a value from a multi-valued property. In this case, path specifies a SCIM filter identifying the value to remove:

```
{
  "op" : "remove",
  "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --remove je-property:je.cleaner.adjustUtilization=false
```

- A second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify, and a SCIM filter whose attribute is value:

```
{
  "op" : "remove",
  "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --remove je-property:je.nodeMaxEntries=32
```

- An option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value:

```
{
  "op" : "remove",
  "path" : "id2childrenIndexEntryLimit"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --reset id2childrenIndexEntryLimit
```

The following is the full example request. The API responds with the entire modified configuration object, which may include a SCIM extension attribute `urn:unboundid:schemas:configuration:messages` containing additional instructions:

```
PATCH /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json

{
  "schemas" : [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  } ]
}
```

Example response:

```
{
 "schemas": [
   "urn:unboundid:schemas:configuration:2.0:backend:local-db"
 ],
 "id": "userRoot2",
 "meta": {
   "resourceType": "Local DB Backend",
   "location": "http://example.com:5033/config/backends/userRoot2"
 },
 "backendID": "userRoot2",
 "backgroundPrime": "false",
 "backupFilePermissions": "700",
 "baseDN": [
   "dc=example2,dc=com"
 ],
 "checkpointOnCloseCount": "2",
 "cleanerThreadWaitTime": "120000",
 "compressEntries": "false",
 "continuePrimeAfterCacheFull": "false",
 "dbBackgroundSyncInterval": "1 s",
 "dbCachePercent": "10",
 "dbCacheSize": "0 b",
 "dbCheckpointerBytesInterval": "20 mb",
 "dbCheckpointerHighPriority": "false",
 "dbCheckpointerWakeupInterval": "1 m",
 "dbCleanOnExplicitGC": "false",
 "dbCleanerMinUtilization": "75",
 "dbCompactKeyPrefixes": "true",
 "dbDirectory": "db",
 "dbDirectoryPermissions": "700",
 "dbEvictorCriticalPercentage": "0",
 "dbEvictorLruOnly": "false",
 "dbEvictorNodesPerScan": "10",
 "dbFileCacheSize": "1000",
 "dbImportCachePercent": "60",
 "dbLogFileMax": "50 mb",
 "dbLoggingFileHandlerOn": "true",
 "dbLoggingLevel": "CONFIG",
 "dbNumCleanerThreads": "0",
 "dbNumLockTables": "0",
 "dbRunCleaner": "true",
 "dbTxnNoSync": "false",
 "dbTxnWriteNoSync": "true",
 "dbUseThreadLocalHandles": "true",
 "deadlockRetryLimit": "10",
 "defaultCacheMode": "cache-keys-and-values",
 "defaultTxnMaxLockTimeout": "10 s",
 "defaultTxnMinLockTimeout": "10 s",
 "description": "123",  "enabled": "false",
 "explodedIndexEntryThreshold": "4000",
 "exportThreadCount": "0",
 "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
 "externalTxnDefaultMaxLockTimeout": "100 ms",
 "externalTxnDefaultMinLockTimeout": "100 ms",
 "externalTxnDefaultRetryAttempts": "2",
 "hashEntries": "false",
 "importTempDirectory": "import-tmp",
 "importThreadCount": "16",
 "indexEntryLimit": "4000",
 "isPrivateBackend": "false",
 "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
 "jeProperty": [  "\"je.env.backgroundReadLimit=0\""
 ],
 "numRecentChanges": "50000",
 "offlineProcessDatabaseOpenTimeout": "1 h",
 "primeAllIndexes": "true",
 "primeMethod": [
   "none"
 ],
 "primeThreadCount": "2",
 "primeTimeLimit": "0 ms",
 "processFiltersWithUndefinedAttributeTypes": "false",
 "returnUnavailableForUntrustedIndex": "true",
 "returnUnavailableWhenDisabled": "true",
 "setDegradedAlertForUntrustedIndex": "true",
 "setDegradedAlertWhenDisabled": "true",
 "subtreeDeleteBatchSize": "5000",
 "subtreeDeleteSizeLimit": "5000",
 "uncachedId2entryCacheMode": "cache-keys-only",
 "writabilityMode": "enabled",
 "urn:unboundid:schemas:configuration:messages:2.0": {
   "requiredActions": [
     {
       "property": "jeProperty",
       "type": "componentRestart",
       "synopsis": "In order for this modification to take effect,
          the component must be restarted, either by disabling and
          re-enabling it, or by restarting the server"
     },
     {
       "property": "id2childrenIndexEntryLimit",
       "type": "other",
       "synopsis": "If this limit is increased, then the contents
          of the backend must be exported to LDIF and re-imported to
          allow the new limit to be used for any id2children keys
          that had already hit the previous limit."
     }
   ]
 }
}
```

Configuration API Paths

The Configuration API is available under the /config path. A full listing of supported sub-paths is available by accessing the base `/config/ResourceTypes` endpoint:

```
GET /config/ResourceTypes
Host: example.com:5033
Accept: application/scim+json
```

Sample response (abbreviated):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "dsee-compat-access-control-handler",
      "name": "DSEE Compat Access Control Handler",
      "description": "The DSEE Compat Access Control
              Handler provides an implementation that uses syntax
              compatible with the Sun Java System Directory Server
              Enterprise Edition access control handler.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/dsee-compat-
access-control-handler"
      }
    },
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "access-control-handler",
      "name": "Access Control Handler",
      "description": "Access Control Handlers manage the
              application-wide access control. The server's access
              control handler is defined through an extensible
              interface, so that alternate implementations can be created.
              Only one access control handler may be active in the server
                       at any given time.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/access-
control-handler"
      }
    },
    {
 ...
```

The response's `endpoint` elements enumerate all available sub-paths. The path `/config/access-control-handler` in the example can be used to get a list of existing access control handlers, and create new ones. A path containing an object name such as `/config/backends/{backendName}`, where `{backendName}` corresponds to an existing backend (such as `userRoot`) can be used to obtain an object's properties, update the properties, or delete the object.

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted nor created. These paths can be differentiated from others by their singular, rather than plural, relation name (for example `global-configuration`).

Sorting and Filtering Objects

The Configuration API supports SCIM parameters for filter, sorting, and pagination. Search operations can specify a SCIM filter used to narrow the number of elements returned. See the SCIM specification for the full set of operations for SCIM filters. Clients can also specify sort parameters, or paging parameters. Include or exclude attributes can be specified in both get and list operations.

GET Parameter	Description
filter	Values can be simple SCIM filters such as `id eq "userRoot"` or compound filters like `meta.resourceType eq "Local DB Backend"` and `baseDn co "dc=exmple,dc=com"`.
sortBy	Specifies a property value by which to sort.
sortOrder	Specifies either `ascending` or `descending` alphabetical order.
startIndex	1-based index of the first result to return.
count	Indicates the number of results per page.

Updating Properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH. With PUT, the server computes the differences between the object in the request with the current version in the server, and performs modifications where necessary. The server will never remove attributes that are not specified in the request. The API responds with the entire modified object.

Request:

```
PUT /config/backends/userRoot
```

```
Host: example.com:5033
Accept: application/scim+json
{
  "description" : "A new description."
}
```

Response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode":
  "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "abc",
  "enabled": "true",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior":
  "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "true",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "numRecentChanges": "50000",  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative Actions

Updating a property may require an administrative action before the change can take effect. If so, the server will return `200 Success`, and any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

For example, changing the `jeProperty` of a backend will result in the following:

```
"urn:unboundid:schemas:configuration:messages:2.0": {
    "required-actions": [
      {
       "property": "baseContextPath",
       "type": ""componentRestart",
       "synopsis": "In order for this modification to take effect, the
 component
                  must be restarted, either by disabling and re-enabling it,
 or
                  by restarting the server"
      },
      {
        "property": {
        "type": "other",
        "synopsis": "If this limit is increased, then the
                 contents of the backend must be exported to LDIF
                 and re-imported to allow the new limit to be used
                 for any id2children keys that had already hit the
                 previous limit."
      }
    ]
  }
```

Updating Servers and Server Groups

Servers can be configured as part of a server group, so that configuration changes that are applied to a single server, are then applied to all servers in a group. When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query attribute. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `single-server` or `server-group` can be specified. For example:

```
http://localhost:8082/config/backends/userRoot?applyChangeTo=single-server
```

Configuration API Responses

Clients of the API should examine the HTTP response code in order to determine the success or failure of a request. The following are response codes and their meanings:

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, or object properties, administrative actions.
204 No Content	The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation.	None.
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message.
401 Unauthorized	User authentication is required. Some user agents such as browsers may respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid.	None.
403 Forbidden	The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted.	None.
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message.
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists, or an attempt was made to delete an object that is referred to by another object.	Error summary and optional message.
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None.
500 Server Error	The server encountered an unexpected error. Please report server errors to customer support.	Error summary and optional message.

An application that uses the Configuration API should limit dependencies on particular text appearing in error message content. These messages may change, and their presence may depend on server configuration. Use the HTTP return code and the context of the request to create a client error message. The following is an example encoded error message:

```
{
 "schemas": [
   "urn:ietf:params:scim:api:messages:2.0:Error"
 ],
 "status": 404,
 "scimType": null,
 "detail": "The Local DB Index does not exist."
}
```

Configure the Server Using the Administrative Console

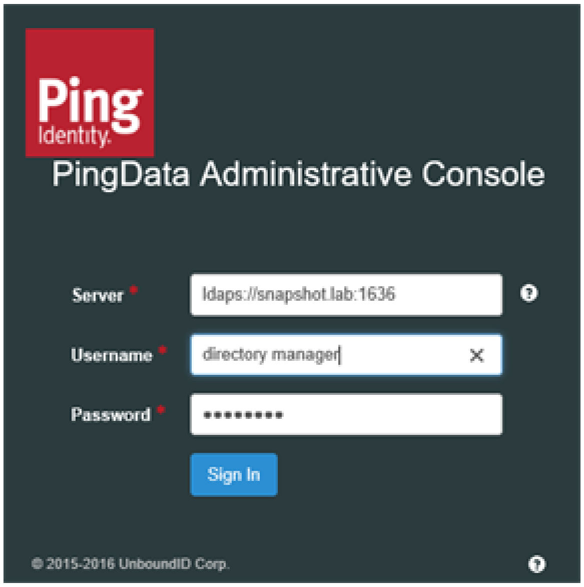
The PingDirectory Server provides a Administrative Console for server configuration and monitoring that has the same functionality as that of the `dsconfig` command. When logging on to the Administrative Console, the console does not persistently store any credentials for authenticating to the Directory Server but uses the credentials provided by the user when logging in. When managing multiple directory server instances, the provided credentials must be valid for each instance.

To Log onto the Administrative Console

To log into the console, enter a fully qualified DN (for example, `cn=admin2,cn=Topology Admin Users,cn=Topology,cn=config`). See *Configuring a Global Administrator* for instructions.

1. Start the Directory Server.

   ```
   $ bin/start-server
   ```
2. Open a browser to http://server-name:389/console/login.
3. Enter the root user DN and password, then click **Login**.



Server * ldaps://snapshot.lab:1636

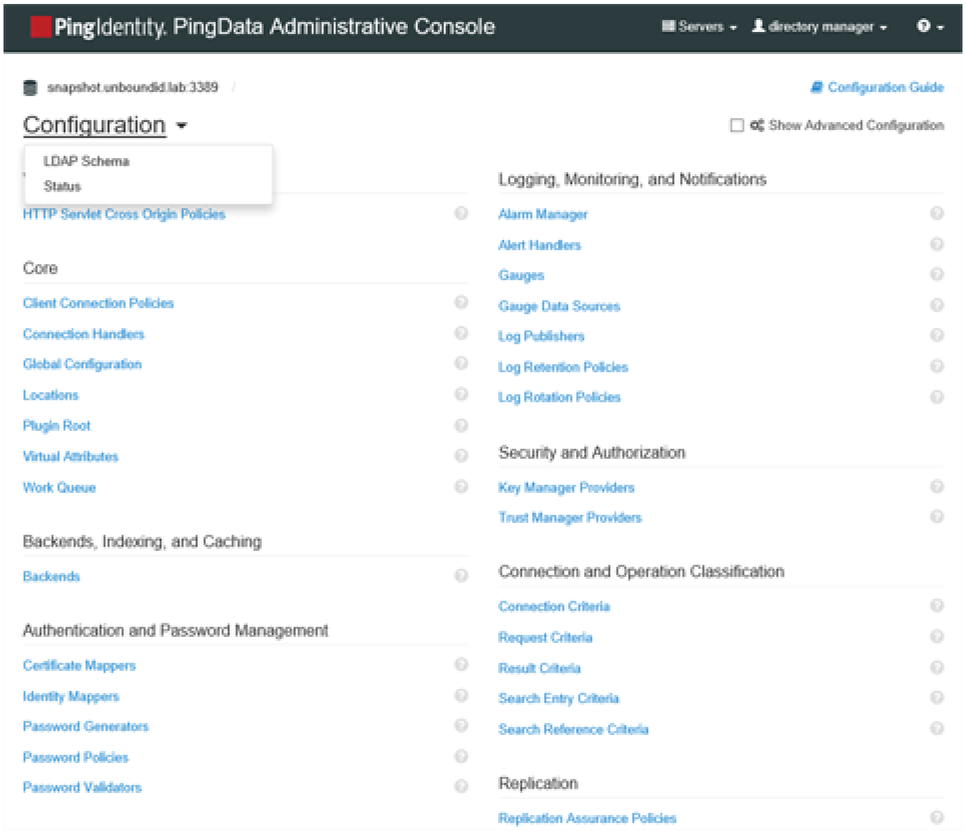
Username * directory manager

Password * ********

The console does not persistently store any credentials for accessing the Directory Server. Instead, it uses the credentials provided by the user when logging into the console. When managing multiple directory server instances, the provided credentials must be valid at each instance.

To Configure the Server Using the Console

1. Log into the Administrative Console. Click **Configuration** to open the **Configuration** menu, and then click **Backends**.


PingIdentity. PingData Administrative Console
Servers
directory manager
snapshot.unboundid.lab:3389
Configuration Guide
Configuration
□ Show Advanced Configuration
LDAP Schema
Status

HTTP Servlet Cross Origin Policies

Core

- Client Connection Policies
- Connection Handlers
- Global Configuration
- Locations
- Plugin Root
- Virtual Attributes
- Work Queue

Backends, Indexing, and Caching

- Backends

Authentication and Password Management

- Certificate Mappers
- Identity Mappers
- Password Generators
- Password Policies
- Password Validators

Logging, Monitoring, and Notifications

- Alarm Manager
- Alert Handlers
- Gauges
- Gauge Data Sources
- Log Publishers
- Log Retention Policies
- Log Rotation Policies

Security and Authorization

- Key Manager Providers
- Trust Manager Providers

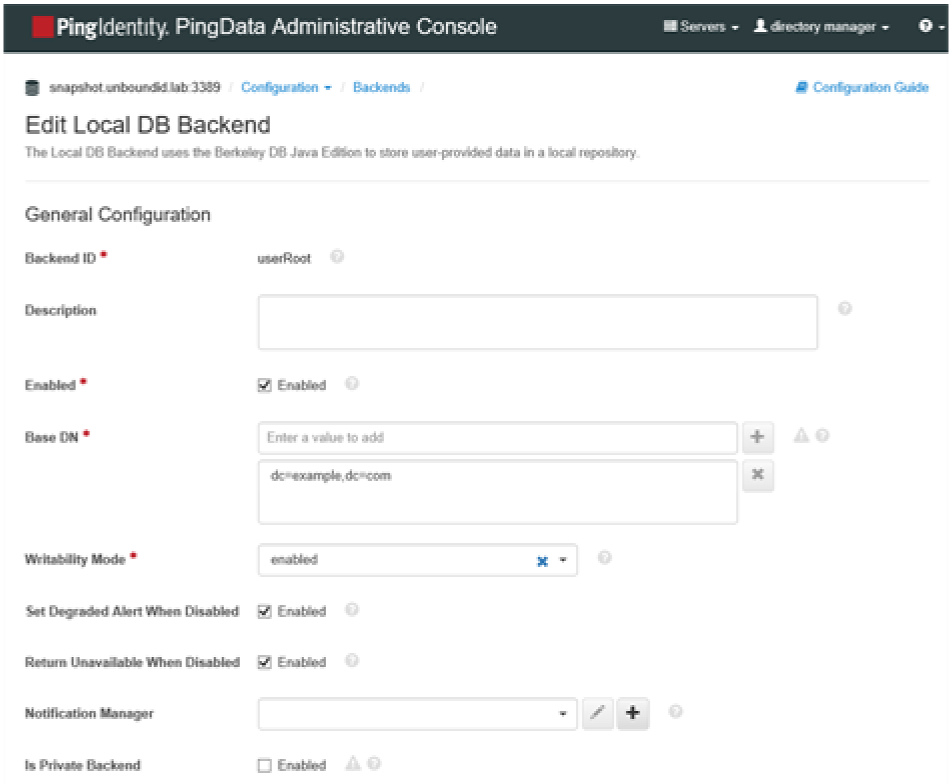
Connection and Operation Classification

- Connection Criteria
- Request Criteria
- Result Criteria
- Search Entry Criteria
- Search Reference Criteria

Replication

- Replication Assurance Policies

2. At the top of the Administrative Console page, click **Show Advanced Properties**. Click **Backends**. For this example, click **userRoot**.


PingIdentity. PingData Administrative Console
Servers
directory manager
snapshot.unboundid.lab:3389 / Configuration / Backends /
Configuration Guide

Edit Local DB Backend

The Local DB Backend uses the Berkeley DB Java Edition to store user-provided data in a local repository.

General Configuration

Backend ID *	userRoot
Description	
Enabled *	☑ Enabled
Base DN *	Enter a value to add dc=example,dc=com
Writability Mode *	enabled
Set Degraded Alert When Disabled	☑ Enabled
Return Unavailable When Disabled	☑ Enabled
Notification Manager	
Is Private Backend	□ Enabled

3. Change or enter values in the `userRoot` configuration. For this example, change the `DB Cache Percent` value to 40, and then click **Save**.

Generating a Summary of Configuration Components

The Directory Server provides a `config-diff` tool that generates a summary of the configuration in a local or remote directory server instance. The tool is useful when comparing configuration settings on the directory server instance when troubleshooting issues or when verifying configuration settings on newly-added servers to your network. The tool can interact with the local configuration regardless of whether the server is running or not.

Run the `config-diff` --help option to view other available tool options.

To Generate a Summary of Configuration Components

- Run the `config-diff` tool to generate a summary of the configuration components on the directory server instance. The following command runs a summary on a local online server.

```
$ bin/config-diff
```

- The following example compares the current configuration of the local server to the baseline, pre-installation configuration, ignoring any changes that could be made by the installer, and writes the output to the `configuration-steps.dsconfig` file. This provides a script that can be used to configure a newly installed server identically to the local server:

```
$ bin/config-diff --sourceLocal \
  --sourceBaseline \
  --targetLocal \
  --exclude differs-after-install \
  --outputFile configuration-steps.dsconfig
```

About Root User, Administrator, and Global Administrators

The Directory Server provides three different classes of administrator accounts: root user, administrator, and global administrator. The root user is the LDAP-equivalent of a UNIX super-user account and inherits its privileges from the default root user privilege set (see *Default Root Privileges*). The root user "account" is an entry that is stored in the server’s configuration under the `cn=Root DNs,cn=config` and bypasses access control evaluation, and can be created manually, or with the `dsconfig` tool. This account has full access to the entire set of data in the Directory Information Tree (DIT) as well as full access to the server configuration and its operations. One important difference between other vendors’ servers and the Directory Server’s implementation is that the root user’s rights are granted through a set of privileges. This allows the Directory Server to have multiple root users on its system if desired; however, the normal practice is to set up administrator user entries. Also, by default, the Root User has no resource limits.

The administrator user can have a full set of root user privileges but often has a subset of these privileges to limit the accessible functions that can be performed. The administrators entries typically have limited access to the entire set of data in the directory information tree (DIT), which is controlled by access control instructions. These entries reside in the backend configuration (for example, `uid=admin,dc=example,dc=com`) and are replicated between servers in a replication topology. In some cases, administrator user accounts may be unavailable when the server enters lockdown mode unless the administrator is given the lock-down mode privilege.

A global administrator is primarily responsible for managing configuration server groups. A configuration server group is an administration domain that allows you to synchronize configuration changes to one or all of the servers in the group. For example, you can set up a group when configuring a replication topology, where configuration changes to one server can be applied to all of the servers at one time. Global Administrator entries are stored in the `cn=Topology Admin Users,cn=Topology,cn=config` backend are always mirrored across servers in a replication topology. These users can be assigned privileges like other admin users but are typically used to manage the data under `cn=Topology,cn=config`.

Managing Root Users Accounts

The PingDirectory Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file (for example, under `cn=Root DNs,cn=config`). The root user is the LDAP-equivalent of a UNIX super-user account and inherits its read-write privileges from the default root privilege set. Root users can be created and updated with the `dsconfig` tool. Root user entries are stored in the server's configuration.The following is a sample command to create a new root user:

```
bin/dsconfig create-root-dn-user --user-name "Joanne Smith" \
  --set last-name:Smith \
  --set first-name:Joanne \
  --set user-id:jsmith \
  --set 'email-address:jsmith@example.com' \
  --set mobile-telephone-number:8889997777 \
  --set home-telephone-number:5556667777 \
  --set work-telephone-number:4445556666
```

To limit full access to all of the Directory Server, create separate administrator accounts with limited privileges so that you can identify the administrator responsible for a particular change. Having separate user accounts for each administrator also makes it possible to enable password policy functionality (such as password expiration, password history, and requiring secure authentication) for each administrator.

Default Root Privileges

The PingDirectory Server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments.

Note: Creating restricted root user accounts requires assigning privileges and necessary access controls for actions on specific data or backends. Access controls are determined by how the directory is configured and the structure of your data. See Chapter 16: Managing Access Controls for more information.

The following set of root privileges are available to each root user DN:

Table 4: Default Root Privileges

Privilege	Description
audit-data-security	Allows the associated user to execute data security auditing tasks.
backend-backup	Allows the user to perform backend backup operations.
backend-restore	Allows the user to perform backend restore operations.
bypass-acl	Allows the user to bypass access control evaluation.
config-read	Allows the user to read the server configuration.
config-write	Allows the user to update the server configuration.
disconnect-client	Allows the user to terminate arbitrary client connections.
ldif-export	Allows the user to perform LDIF export operations.
ldif-import	Allows the user to perform LDIF import operations.
lockdown-mode	Allows the user to request a server lockdown.
manage-topology	Allows the user to modify topology setting.

Privilege	Description
metrics-read	Allows the user to read server metrics.
modify-acl	Allows the user to modify access control rules.
password-reset	Allows the user to reset user passwords but not their own. The user must also have privileges granted by access control to write the user password to the target entry.
permit-get-password-policy-state-issues	Allows the user to access password policy state issues.
privilege-change	Allows the user to change the set of privileges for a specific user, or to change the set of privileges automatically assigned to a root user.
server-restart	Allows the user to request a server restart.
server-shutdown	Allows the user to request a server shutdown.
soft-delete-read	Allows the user access to soft-deleted entries.
stream-values	Allows the user to perform a stream values extended operation that obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
third-party-task	Allows the associated user to invoke tasks created by third-party developers.
unindexed-search	Allows the user to perform an unindexed search in the Oracle Berkeley DB Java Edition backend.
update-schema	Allows the user to update the server schema.
use-admin-session	Allows the associated user to use an administrative session to request that operations be processed using a dedicated pool of worker threads.

The Directory Server provides other privileges that are not assigned to the root user DN by default but can be added using the `ldapmodify` tool (see Modifying Individual Root User Privileges) for more information.

Table 5: Other Available Privileges

Privilege	Description
bypass-pw-policy	Allows the associated user bypass password policy rules and restrictions.
bypass-read-aci	Allows the associated user to bypass access control checks performed by the server for bind, compare, and search operations. Access control evaluation may still be enforced for other types of operations.
jmx-notify	Allows the associated user to subscribe to receive JMX notifications.
jmx-read	Allows the associated user to perform JMX read operations.
jmx-write	Allows the associated user to perform JMX write operations.
permit-externally-processed-authentication	Allows the associated user accept externally processed authentication.
permit-proxied-mschapv2-details	Allows the associated user to permit MS-CHAP V2 handshake protocol.
proxied-auth	Allows the associated user to accept proxied authorization.

Configuring Administrator Accounts

An administrator account is any account in the user backend that is assigned one or more privileges, or given access to read and write operations beyond that of a normal user entry. The privilege mechanism is the same as that used for Root DN accounts and allows individual privileges to be assigned to an administrator entry.

Typically, administrator user entries are controlled by access control evaluation to limit access to the entire set of data in the Directory Information Tree (DIT). Fine-grained read and write access can be granted using the access control definitions available via the `aci` attribute. Administrator entries reside in the backend configuration (for example, `uid=admin,dc=example,dc=com`) and are replicated between servers in a replication topology.

The following examples show how to configure administrator accounts. The first procedure shows how to set up a single, generic `uid=admin,dc=example,dc=com` account with limited privileges. Note that if you generated sample data at install, you can view an example `uid=admin` entry using `ldapsearch`. The second example shows a more realistic example, where the user is part of the Administrators group. Note that both examples are based on a simple DIT. Actual deployment cases depends on your schema.

To Set Up a Single Administrator Account

1. Create an LDIF file with an example Administrator entry.

```
dn: uid=admin,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: Admin
uid: admin
cn: Admin User
sn: User
userPassword: password
```

2. Then add the entry using the `ldapmodify` tool.

```
$ bin/ldapmodify --defaultAdd --filename admin.ldif
```

3. Create another LDIF file to add the access control instruction (ACI) to the root suffix, or base DN to give full access to the new administrator. The ACI grants full access to all user attributes, but not to operational attributes. If you want to grant access to operational attributes as well as user attributes, use `(targetattr = "*||+")` in the access control instruction.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

4. Then add the entry using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename admin.ldif
```

5. Verify the additions using `ldapsearch`. The first command searches for the entry that contains `uid=Admin` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including access control instructions, associated with the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=admin)"

$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope base
 "(objectclass=*)" "+"
```

6. Add specific privileges to the Admin account. In this example, add the `password-reset` privilege to the admin account from the command line. After typing the privileges, press **CTRL-D** to process the modify operation.

```
$ bin/ldapmodify
dn: uid=admin,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset
```

```
Processing MODIFY request for uid=admin,dc=example,dc=com
MODIFY operation successful for DN uid=admin,dc=example,dc=com
```

7. Assign a password policy for the Admin account. For example, create an "Admin Password Policy", then add the password policy to the account.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Admin Password Policy" \
  --set "description:Password policy for administrators" \
  --set password-attribute:userpassword \
  --set "default-password-storage-scheme:Salted 256-bit SHA-2" \
  --set password-change-requires-current-password:true \
  --set force-change-on-reset:true \
  --set "max-password-age:25w 5d" \
  --set grace-login-count:3
  --no-prompt
```

8. Apply the password policy to the account. In this example, the password policy is being added from the command line. The following `ldapmodify` command should be executed with a bind DN that has sufficient rights, such as a Root DN.

```
$ bin/ldapmodify
dn: uid=admin,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Admin Password Policy,cn=Password
 Policies,cn=config
```

To Change the Administrator Password

Root users are governed by the Root Password Policy and by default, their passwords never expire. However, if a root user password must be changed, use the `ldappasswordmodify` tool.

1. Open a text editor and create a text file containing the new password. In this example, name the file `rootuser.txt`.

```
$ echo password > rootuser.txt
```

2. Use `ldappasswordmodify` to change the root user’s password.

```
$ bin/ldappasswordmodify --port 1389 --bindDN "cn=Directory Manager"\
  --bindPassword secret --newPasswordFile rootuser.txt
```

3. Remove the text file.

```
$ rm rootuser.txt
```

To Set Up an Administrator Group

The following example shows how to set up a group of administrators that have access rights to the whole Directory Server. The example uses a static group using the `GroupOfUniqueNames` object class.

1. Create an LDIF file with an example Administrator group, and save it as `admin-group.ldif`.

```
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
```

```
ou: Groups

dn: cn=Dir Admins,ou=Groups,dc=example,dc=com
objectClass: groupofuniquenames
objectClass: top
uniqueMember: uid=user.0, ou=People, dc=example,dc=com
uniqueMember: uid=user.1, ou=People, dc=example,dc=com
cn: Dir Admins
ou: Groups
```

2. Then, add the entries using the `ldapmodify` tool.

```
$ bin/ldapmodify --defaultAdd --filename admin-group.ldif
```

3. Create another LDIF file to add the access control instruction (ACI) to the root suffix, or base DN to provide full access to the Directory Server to the new administrator. Save the file as `admin-aci.ldif`.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com")
  (targetattr != "aci")
  (version 3.0; acl "allow all Admin group";
    allow(all) groupdn = "ldap:///cn=Dir
 Admins,ou=Groups,dc=example,dc=com";)
```

4. Then, add the ACI using the `ldapmodify` tool:

```
$ bin/ldapmodify --filename admin-aci.ldif
```

5. Verify the additions using `ldapsearch`. The first command searches for the entry that contains `cn=Dir Admins` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including access control instructions, associated with the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(cn=Dir Admins)"

$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope base \
  "(objectclass=*)" "+"
```

6. Add specific privileges to each Admin account using an LDIF file, saved as `admin-priv.ldif`. In this example, add the `password-reset` privilege to the `user.0` admin account from the command line. Add the privilege using the `ldapmodify` tool. Repeat the process for the other administrators configured in the Admin group.

```
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset

$ bin/ldapmodify --filename admin-priv.ldif
```

```
Processing MODIFY request for uid=user.0,dc=example,dc=com
MODIFY operation successful for DN uid=user.0,dc=example,dc=com
```

7. Assign a password policy for the Admin account using an LDIF file, saved as `admin-pwd-policy.ldif`. For example, create an "Admin Password Policy", then add the password policy to the account. Apply the password policy to the account using the `ldapmodify` tool.

```
dn: uid=user.0,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Admin Password Policy,cn=Password
 Policies,cn=config

$ bin/ldapmodify --filename admin-pwd-policy.ldif
```

Configuring a Global Administrator

A global administrator is created when replication is enabled, and is responsible for managing configuration server groups. A configuration server group is an administration domain that allows you to synchronize configuration changes to one or all of the servers in the group. For example, you can set up a group when configuring a replication topology, where configuration changes to one server can be applied to all of the servers at a time.

Global Administrator(s) are stored in the topology registry. These entries are always mirrored between servers in a topology. Global Administrators can be assigned privileges like other admin users but are typically used to manage the data under `cn=topology,cn=config` and `cn=config`. You can create new global administrators and remove existing global administrators using the `dsconfig` tool. The global administrator entries are located in the `cn=Topology Admin User, cn=topology,cn=config` branch.

To Create a Global Administrator

1. Use `dsconfig` to create a new global administrator.

```
$ bin/dsconfig create-topology-admin-user \
  --user-name admin2 \
  --set alternate-bind-dn:cn=admin2 \
  --set password:rootPassword
```

2. To verify the creation of the new administrator, use the `list-topology-admin-users` subcommand with `dsconfig`.

```
$ bin/dsconfig list-topology-admin-users
Topology Admin User : Type
____________________:_______
admin               : generic
admin2              : generic
```

To Remove a Global Administrator

1. Use `dsconfig` to delete an existing global administrator.

```
$ bin/dsconfig delete-topology-admin-user --user-name admin2
```

2. To verify the deletion of the global administrator, use the list-topology-admin-users option with `dsconfig`.

```
$ bin/dsconfig list-topology-admin-users
Topology Admin User : Type
____________________:______
admin               : generic
```

Configuring Server Groups

The PingDirectory Server provides a mechanism for setting up administrative domains that synchronize configuration changes among servers in a server group. After you have set up a server group, you can make an update on one server using `dsconfig`, then you can apply the change to the other servers in the group using the --applyChangeTo server-group option of the `dsconfig` non-interactive command. If you want to apply the change to one server in the group, use the --applyChangeTo single-server option. When using `dsconfig` in interactive command-line mode, you will be asked if you want to apply the change to a single server or to all servers in the server group.

About the Server Group Example

You can create an administrative server group using the `dsconfig` tool. The general process is to create a group, add servers to the group, and then set a global configuration property to use the server group. If you are configuring a replication topology, then you must configure the replicas to be in a server group as outlined in Replication Configuration.

The following example procedure adds three Directory Server instances into the server group labelled "group-one".

To Create a Server Group

1. Create a group called “group-one” using `dsconfig`.

```
$ bin/dsconfig create-server-group --group-name group-one
```

2. Add any directory server to the server group. If you have set up replication between a set of servers, these server entries will have already been created by the `dsreplication` enable command.

```
$ bin/dsconfig set-server-group-prop --group-name group-one --set
 member:server1

$ bin/dsconfig set-server-group-prop --group-name group-one --set
 member:server2

$ bin/dsconfig set-server-group-prop --group-name group-one --set
 member:server3
```

3. Set a global configuration property for each of the servers that should share changes in this group.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:group-one
```

4. Test the server group. In this example, enable the log publisher for each directory server in the group, server-group, by using the --applyChangeTo server-group option.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true \
  --applyChangeTo server-group
```

5. View the property on the first directory server instance.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : true
```

6. Repeat step 5 on the second and third directory server instance.
7. Test the server group by disabling the log publisher on the first directory server instance by using the --applyChangeTo single-server.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:disabled \
  --applyChangeTo single-server
```

8. View the property on the first directory server instance. The first directory server instance should be disabled.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : false
```

9. View the property on the second directory server instance. Repeat this step on the third directory server instance to verify that the property is still enabled on that server.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : true
```

Configuring Client Connection Policies

Client connection policies help distinguish what portions of the DIT the client can access. They also enforce restrictions on what clients can do in the server. A client connection policy specifies criteria for membership based on information about the client connection, including client address, protocol, communication security, and authentication state and identity. The client connection policy, however, does not control membership based on the type of request being made.

Every client connection is associated with exactly one client connection policy at any given time, which is assigned to the client when the connection is established. The choice of which client connection policy to use will be reevaluated when the client attempts a bind to change its authentication state or uses the StartTLS extended operation to convert an insecure connection to a secure one. Any changes you make to the client connection policy do not apply to existing connections. The changes only apply to new connections.

Client connections are always unauthenticated when they are first established. If you plan to configure a policy based on authentication, you must define at least one client connection policy with criteria that match unauthenticated connections.

Once a client has been assigned to a policy, you can determine what operations they can perform. For example, your policy might allow only SASL bind operations. Client connection policies are also associated with one or more subtree views, which determine the portions of the DIT a particular client can access. For example, you might configure a policy that prevents users connecting over the extranet from accessing configuration information. The client connection policy is evaluated in addition to access control, so even a root user connecting over the extranet would not have access to the configuration information.

Understanding the Client Connection Policy

Client connection policies are based on two things:

- **Connection criteria**. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used in other instances when the server needs to perform matching based on connection-level properties, such as filtered logging. A single connection can match multiple connection criteria definitions.
- **Evaluation order index**. If multiple client connection policies are defined in the server, then each of them must have a unique value for the **evaluation-order-index** property. The client connection policies are evaluated in order of ascending evaluation order index. If a client connection does not match the criteria for any defined client connection policy, then that connection will be terminated.

If the connection policy matches a connection, then the connection is assigned to that policy and no further evaluation occurs. If, after evaluating all of the defined client connection policies, no match is found, the connection is terminated.

When a Client Connection Policy is Assigned

A client connection policy can be associated with a client connection at the following times:

- When the connection is initially established. This association occurs exactly once for each client connection.
- After completing processing for a StartTLS operation. This association occurs at most once for a client connection, because StartTLS cannot be used more than once on a particular connection. You also may not stop using StartTLS while keeping the connection active.

- After completing processing for a bind operation. This association occurs zero or more times for a client connection, because the bind request can be processed many times on a given connection.

StartTLS and bind requests will be subject to whatever constraints are defined for the client connection policy that is associated with the client connection at the time that the request is received. Once they have completed, then subsequent operations will be subject to the constraints of the new client connection policy assigned to that client connection. This policy may or may not be the same client connection policy that was associated with the connection before the operation was processed. That is, any policy changes do not apply to existing connections and will be applicable when the client reconnects.

All other types of operations will be subject to whatever constraints are defined for the client connection policy used by the client connection at the time that the request is received. The client connection policy assigned to a connection never changes as a result of processing any operation other than a bind or StartTLS. So, the server will not re-evaluate the client connection policy for the connection in the course of processing an operation. For example, the client connection policy will never be re-evaluated for a search operation.

Restricting the Type of Search Filter Used by Clients

You can restrict the types of search filters that a given client may be allowed to use to prevent the use of potentially expensive filters, like range or substring searches. You can use the `allowed-filter-type` property to provide a list of filter types that may be included in the search requests from clients associated with the client connection policy. This setting will only be used if search is included in the set of allowed operation types. This restriction will only be applied to searches with a scope other than `baseObject`, such as searches with a scope of `singleLevel`, `wholeSubtree`, or `subordinateSubtree`.

The `minimum-substring-length` property can be used to specify the minimum number of non-wildcard characters in a substring filter. Any attempt to use a substring search with an element containing fewer than this number of bytes will be rejected. For example, the server can be configured to reject filters like `"(cn=a*)"` and `"(cn=ab*)"`, but to allow `"(cn=abcde*)"`. This property setting will only be used if search is included in the set of allowed operation types and at least one of `sub-initial`, `sub-any`, or `sub-final` is included in the set of allowed filter types.

There are two primary benefits to enforcing a minimum substring length:

- Allowing very short substrings can require the server to perform more expensive processing. The search requires a lot more server effort to assemble a candidate entry list for short substrings because the server has to examine a lot more index keys.
- Allowing very short substrings makes it easier for a client to put together a series of requests to retrieve all the data from the server (a process known as "trawling"). If a malicious user wants to obtain all the data from the server, then it is easier to issue 26 requests like `"(cn=a*)"`, `"(cn=b*)"`, `"(cn=c*)"`, ..., `"(cn=z*)"` than if the user is required to do something like `"(cn=aaaaa*)"`, `"(cn=aaaab*)"`, `"(cn=aaaac*)"`, ..., `"(cn=zzzzz*)"`.

Setting Resource Limits

Client connection policies can specify resource limits, helping to ensure that no single client monopolizes server resources. You can limit the total number of connections to a server from a particular client or from clients that match specified criteria. You can also limit the duration of the connection.

A client connection policy may only be used to enforce additional restrictions on a client connection. You can never use it to grant a client capabilities that it would not otherwise have.

Any change to any of these new configuration properties will only impact client connections that are assigned to the client connection policy after the change is made. Any connection associated with the client connection policy before the configuration change was made will continue to be subject to the configuration that was in place at the time it was associated with that policy.

Table 6: Resource Limiting Properties

Property	Description
maximum-concurrent-connections	Specifies the maximum number of client connections that can be associated with that client connection policy at any given time. The default value of zero indicates that no limit will be enforced. If the server already has the maximum number of connections associated with a client connection policy, then any attempt to associate another connection with that policy (e.g., newly-established connections or an existing connection that has done something to change its client connection policy, such as perform a bind or StartTLS operation) will cause that connection to be terminated.
terminate-connection	Specifies that any client connection for which the client connection policy is selected (whether it is a new connection or an existing connection that is assigned to the client connection policy after performing a bind or StartTLS operation) will be immediately terminated. This property can be used to define criteria for connections that you do not want to be allowed to communicate with the Directory Server.
maximum-connection-duration	Specifies the maximum length of time that a connection associated with the client connection policy can remain established to the Directory Server, regardless of the amount of activity on that connection. A value of "0 seconds" (default) indicates that no limit will be enforced. If a connection associated with the client connection policy has been established for longer than this time, then it will be terminated.
maximum-idle-connection-duration	Specifies the maximum length of time that a connection associated with the client connection policy can remain established with the Directory Server without any requests in progress. A value of "0 seconds" (default) indicates that no additional limit will be enforced on top of whatever idle time limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective idle time limit for any client connection will be the smaller of the `maximum-idle-connection-duration` from the client connection policy and the idle time limit that would otherwise be in effect for that client. This property can be used to apply a further restriction on top of any value that may be enforced by the `idle-time-limit` global configuration property (which defines a default idle time limit for client connections) or the `ds-rlim-idle-time-limit` operational attribute (which may be included in a user entry to override the default idle time limit for that user).
maximum-operation-count-per-connection	Specifies the maximum number of operations that a client associated with the client connection policy will be allowed to request. A value of zero (default) indicates that no limit will be enforced. If a client attempts to request more than this number of operations on the same connection, then that connection will be terminated.
maximum-concurrent-operations-per-connection	Specifies the maximum number of operations that may be active at any time from the same client. This limit is only applicable to clients that use asynchronous operations with multiple outstanding requests at any given time. A value of zero (default) indicates that no limit will be enforced.

Property	Description
	If a client already has the maximum number of outstanding requests in progress and issues a new request, then that request will be delayed and/or rejected based on the value of the `maximum-concurrent-operation-wait-time-before-rejecting` property.
maximum-concurrent-operation-wait-time-before-rejecting	Specifies the maximum length of time that a client connection should allow an outstanding operation to complete if the maximum number of concurrent operations for a connection are already in progress when a new request is received on that connection. A value of “0 seconds” (default) indicates that any new requests received while the maximum number of outstanding requests are already in progress for that connection will be immediately rejected. If an outstanding operation completes before this time expires, then the server may be allowed to process that operation. If the time expires, the new request will be rejected.
maximum-ldap-join-size-limit	Specifies the maximum number of entries that can be directly joined with any individual search result entry. A value of zero indicates that no LDAP join size limit is enforced. The limit can be overridden on a per-user basis using the `ds-rlim-ldap-join-size-limit` operational attribute. The LDAP join size limit is also restricted by the search operation size limit. If a search result entry is joined with more entries than allowed, the join result control will have a "size limit exceeded" (integer value 4) result code.
allowed-request-control	Specifies the OIDs of the request controls that clients associated with the client connection policy will be allowed to use. If any allowed-request-control OIDs are specified, then any request which includes a control not in that set will be rejected. If no `allowed-request-control` values are specified (default), then any control whose OID is not included in the set of denied-request-control values will be allowed.
denied-request-control	Specifies the OIDs of the request controls that clients associated with the client connection policy will not be allowed to use. If there are any `denied-request-control` values, then any request containing a control whose OID is included in that set will be rejected. If there are no `denied-request-control` values (default), then any request control will be allowed if the `allowed-request-control` property is also empty, or only those controls whose OIDs are included in the set of `allowed-request-control` values will be allowed if at least one `allowed-request-control` value is provided.
allowed-filter-type	Specifies the types of components which may be used in filters included in search operations with a non-base scope that are requested by clients associated with the client connection policy. Any non-base scoped search request whose filter contains a component not included in this set will be rejected. The set of possible filter types include: and or not equality sub-initial sub-any

Property	Description
	sub-final greater-or-equal less-or-equal approximate-match extensible-match By default, all filter types will be allowed. Also note that no restriction will be placed on the types of filters which may be used in searches with a base scope.
allow-unindexed-searches	Specifies whether clients associated with the client connection policy will be allowed to request searches which cannot be efficiently processed using the configured set of indexes. Note that clients will still be required to have the `unindexed-search` privilege, so this option will not grant the ability to perform unindexed searches to clients that would not have otherwise had that ability, but it may be used to prevent clients associated with the client connection policy from requesting unindexed searches when they might have otherwise been allowed to do so. By default, this has a value of "true", indicating that any client associated with the client connection policy that has the `unindexed-search` privilege will be allowed to request unindexed searches.
minimum-substring-length	Specifies the minimum number of bytes, which may be present in any sub-Initial, subAny, or subFinal element of a substring search filter component in a search with a non-baseObject scope. A value of one (which is the default) indicates that no limit will be enforced. This property may be used to prevent clients from issuing overly-vague substring searches that may require the Installing the Directory Server to examine too many entries over the course of processing the request.
maximum-search-size-limit	Specifies the maximum number of entries that may be returned from any single search operation requested by a client associated with this client connection policy. Note that this property only specifies a maximum limit and will never increase any limit that may already be in effect for the client via the `size-limit` global configuration property or the `ds-rlim-size-limit` operational attribute. A value of zero (default) indicates that no additional limit will be enforced on top of whatever size limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective maximum size limit for any search operation requested by the client will be the smaller of the size limit from that search request, the `maximum-search-size-limit` from the client connection policy, and the size limit that would otherwise be in effect for that client.

Defining the Operation Rate

You can configure the maximum operation rate for individual client connections as well as collectively for all connections associated with a client connection policy. If the operation rate limit is exceeded, the Directory Server may either reject the operation or terminate the connection. You can define multiple rate limit values, making it possible to fine tune limits for both a long term average operation rate and short term operation bursts. For example, you can define a limit of one thousand operations per second and one million operations per day, which works out to an average of less than twelve operations per second, but with bursts of up to one thousand operations per second.

Rate limit strings should be specified as a maximum count followed by a slash and a duration. The count portion must contain an integer, and may be followed by a multiplier of k (to indicate that the integer should be interpreted as thousands), m (to indicate that the integer should be interpreted as millions), or g (to indicate that the integer should be interpreted as billions). The duration portion must contain a time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), and may be preceded by an integer to specify a quantity for that unit.

For example, the following are valid rate limit strings:

1/s (no more than one operation over a one-second interval)
10K/5h (no more than ten thousand operations over a five-hour interval)
5m/2d (no more than five million operations over a two-day interval)

You can provide time units in many different formats. For example, a unit of seconds can be signified using s, sec, sect, second, and seconds.

Client Connection Policy Deployment Example

In this example scenario, we assume the following:

Two external LDAP clients are allowed to bind to the Directory Server.
Client 1 should be allowed to open only 1 connection to the server.
Client 2 should be allowed to open up to 5 connections to the server.

Defining the Connection Policies

We need to set a per-client connection policy limit on the number of connections that may be associated with a particular client connection policy. We have to define at least two client connection policies, one for each of the two clients. Each policy must have different connection criteria for selecting the policy with which a given client connection should be associated.

Because the criteria is based on authentication, we must create a third client connection policy that applies to unauthenticated clients, because client connections are always unauthenticated as soon as they are established and before they have sent a bind request. Plus, clients are not required to send a bind request as their first operation.

Therefore, we define the following three client connection policies:

Client 1 Connection Policy, which only allows client 1, with an evaluation order index of 1.
Client 2 Connection Policy, which only allows client 2, with an evaluation order index of 2.
Unauthenticated Connection Policy, which allows unauthenticated clients, with an evaluation order index of 3.

We define simple connection criteria for the Client 1 Connection Policy and the Client 2 Connection Policy with the following properties:

The `user-auth-type` must not include none, so that it will only apply to authenticated client connections.
The `included-user-base-dn` should match the bind DN for the target user. This DN may be full DN for the target user, or it may be the base DN for a branch that contains a number of users that you want treated in the same way.

To create more generic criteria that match more than one user, you could list the DNs of each of the users explicitly in the `included-user-base-dn` property. If there is a group that contains all of the pertinent users, then you could instead use the `[all|any|not-all|not-any]-included-user-group-dn` property to apply to all members of that group. If the entries for all of the users match a particular filter, then you could use the `[all|any|not-all|not-any]-included-user-filter` property to match them.

How the Policy is Evaluated

Whenever a connection is established, the server associates the connection with exactly one client connection policy. The server does this by iterating over all of the defined client connection policies in ascending order of the evaluation order index. Policies with a lower evaluation order index value will be examined before those with a higher evaluation order index value. The first policy that the server finds whose criteria match the client connection will be associated with that connection. If no client connection policy is found with criteria matching the connection, then the connection will be terminated.

So, in our example, when a new connection is established, the server first checks the connection criteria associated with the Client 1 Connection Policy because it has the lowest evaluation order index value. If it finds that the criteria do not match the new connection, the server then checks the connection criteria associated with the Client 2 Connection Policy because it has the second lowest evaluation order index. If these criteria do not match, the server finally checks the connection criteria associated with the Unauthenticated Connection Policy, because it has the third lowest evaluation order index. It finds a match, so the client connection is associated with the Unauthenticated Connection Policy.

After the client performs a bind operation to authenticate to the server, then the client connection policies will be re-evaluated. If client 2 performs the bind, then the Client 1 Connection Policy will not match but the Client 2 Connection Policy will, so the connection will be re-associated with that client connection policy. Whenever a connection is associated with a client connection policy, the server will check to see if the maximum number of client connections have already been associated with that policy. If so, then the newly-associated connection will be terminated.

For example, Client 1 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. Client 1 then sends a bind request. The determination of whether the bind operation is allowed is made based on the constraints defined in the Unauthenticated Connection Policy, because it is the client connection policy already assigned to the client connection. Once the bind has completed, then the server will reevaluate the client connection policy against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to the Client 1 Connection Policy.

Next, Client 2 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. When Client 2 sends a bind request, the operation is allowed based on the constraints defined in the Unauthenticated Connection Policy. Once the bind is complete, the client connection is evaluated against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria do not match, so the client 2 connection is evaluated against the connection criteria associated with Client 2 Connection Policy, because it has the next lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to Client 2 Connection Policy.

Client 1 sends a search request. The Client 1 Connection Policy is used to determine whether the search operation should be allowed, because this is the client connection policy assigned to the client connection for client 1. The connection is not reevaluated, before or after processing the search operation.

To Configure a Client Connection Policy Using the Console

1. Open the Administrative Console. Provide a username and password, and then click **Login**.
2. In the **Core Server** section, click **Client Connection Policies**. If you do not see **Client Connection Policies** on the menu, change the Object Types filter to **Standard**.
3. Click **Add New** to add a new policy.
4. Enter a Policy ID. If you want to base your new client connection policy on an existing policy, select it from the **Template** menu.
5. Configure the properties of the client connection policy. To enable the policy, select **Enabled**.
6. Enter the order in which you want the new policy to be evaluated in the **Evaluation Order Index** box, and then click **Continue**. A policy with a lower index is evaluated before a policy with a higher index. The Directory Server uses the first evaluated policy that applies to a client connection.
7. Select the connection criteria that match the client connection for this policy. Click **View and edit** to change the criteria. Click **Select New** to add new criteria. Select the operations allowed for clients that are members of this connection group. Use the **Add and Remove** buttons to make operations available to clients. Specify the extended operations that clients are allowed and denied to use.
8. Enter the type of authorization allowed and the SASL mechanisms that are allowed and denied in response to client requests.
9. Check the **Include Backend Subtree Views** check box if you want to automatically include the subtree views of backends configured in the Directory Server. You can also choose to include and exclude specific base DNs using the appropriate fields.

10. Once you have finished configuring the properties of your client connection policy, click **Confirm then Save** to review the `dsconfig` command equivalent and save your changes. Click **Save Now** to save your changes without first reviewing the `dsconfig` output.

To Configure a Client Connection Policy Using dsconfig

You can configure a client connection policy using the `dsconfig` tool in interactive mode from the command line. You can access the **Client Connection Policy** menu on the **Standard objects** menu.

1. Use the `dsconfig` tool to create and configure a client connection policy. Specify the host name, connection method, port number, and bind DN as described in previous procedures.
2. On the Directory Server main menu, change to the **Standard objects** menu by entering `o` and then entering the number for the Standard menu.
3. On the Directory Server main menu, enter the number associated with **Client Connection Policy**.
4. On the **Client Connection Policy management** menu, type the number corresponding to **Create a new connection policy**.
5. Enter `n` to create a new client connection policy from scratch.
6. Next, enter a name for the new client connection policy.
7. On the **Enabled Property** menu, select true to enable the connection policy.
8. On the **Evaluation-Order Property** menu, type a value between 0 and 2147483647 to set the evaluation order for the policy. A client connection policy with a lower evaluation-order will be evaluated before one with a higher number. For this example, type `9999`.
9. On the **Client Connection Policy** menu, review the configuration. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the client connection policy.

   Any changes that you make to the client connection policy do not apply to existing connections. They will only apply to new connections.

Restricting Server Access Based on Client IP Address

Another common use case is to limit client access to the Directory Server. Two methods are available:

- **Connection Handlers**. You can limit the IP addresses using the LDAP or LDAPS connection handlers. The connection handlers provide an `allowed-client` property and a `denied-client` property. The `allowed-client` property specifies the set of allowable address masks that can establish connections to the handler. The `denied-client` property specifies the set of address masks that are not allowed to establish connections to the handler.
- **Client Connection Policies**. You can take a more fine-grained approach by restricting access by configuring a new Client Connection Policy, then create a new connection criteria and associate it with the connection policy. Connection criteria define sets of criteria for grouping and describing client connections based on a number of properties, including the protocol, client address, connection security, and authentication state for the connection. Each client connection policy may be associated with zero or more Connection Criteria, and server components may use Connection Criteria to indicate which connections should be processed and what kind of processing should be performed (e.g., to select connections and/or operations for filtered logging, or to classify connections for network groups).

To Restrict Server Access Using the Connection Handlers

- Use `dsconfig` to set the `allowed-client` property for the LDAP connection handler. You should specify the address mask for the range of allowable IP addresses that can establish connections to the Directory Server. You should also specify the loopback address, 127.0.0.1, so that you will still be able to configure the server using the `dsconfig` tool on the local host.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --set "allowed-client:10.6.1.*" \
  --set allowed-client:127.0.0.1
```

To Restrict Server Access Using Client Connection Policies

1. Create a simple connection criteria. The following example uses the dsconfig tool in non-interactive mode. It allows only the Directory Server's IP address and loopback to have access.

```
$ bin/dsconfig set-connection-criteria-prop \
  --criteria-name allowed-ip-addrs \
  --add included-client-address:10.6.1.80 \
  --add included-client-address:127.0.0.1
```

2. Assign the criteria to the client connection policy. After you have run the following command, access is denied to remote IP addresses. The Directory Server does not require a restart.

```
$ bin/dsconfig set-client-connection-policy-prop \
  --policy-name new-policy \
  --set connection-criteria:allowed-ip-addrs
```

3. Add a remote IP range to the criteria. For this example, add 10.6.1.*. Access from any remote servers is allowed. The Directory Server does not require a restart.

```
$ bin/dsconfig set-connection-criteria-prop \
  --criteria-name allowed-ip-addrs \
  --add "included-client-address:10.6.1.*"
```

4. To restore default behavior, you can remove the criteria from the connection policy. The Directory Server does not require a restart. Remember to include the LDAP or LDAPS connection parameters (e.g., hostname, port, bindDN, bindPassword) with the dsconfig command.

```
$ bin/dsconfig set-client-connection-policy-prop \
  --policy-name new-policy --remove connection-criteria:allowed-ip-addrs
```

To Automatically Authenticate Clients that Have a Secure Communication Channel

The Directory Server provides an option to automatically authenticate clients that have a secure communication channel (either SSL or StartTLS) and presented their own certificate. This option is disabled by default, but when enabled, the net effect will be as if the client issued a SASL EXTERNAL bind request on that connection.

This option will be ignored if the client connection is already authenticated (e.g., because it is using StartTLS but the client had already performed a bind before the StartTLS request). If the bind attempt fails, then the connection will remain unauthenticated but usable. If the client subsequently sends a bind request on the connection, then it will be processed as normal and any automatic authentication will be destroyed.

- Run the following dsconfig command.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAPS Connection Handler" \
  --set "auto-authenticate-using-client-certificate:true"
```

Securing the Server with Lockdown Mode

The Directory Server provides tools to enter and leave lockdown mode if the server requires a security lockdown. In lockdown mode, only users with the lockdown-mode privilege can perform operations, while those without the privilege are rejected. Root users have this privilege by default; other administrators can be given this privilege.

The Directory Server can be manually placed into lockdown mode to perform some administrative operation while ensuring that other client requests are not allowed to access any data in the server. In addition, some configuration problems (particularly problems that could lead to inadvertent exposure of sensitive information, like an access control rule that cannot be properly parsed) cause the server to place itself in lockdown mode, so that an administrator can manually correct the problem. Lockdown mode does not persist across restarts. The directory server can be taken out of lockdown mode by using either the leave-lockdown-mode tool or by restarting the server. If administrators want to start a server in lockdown mode, they can use the start-server --lockdownMode option.

Any client request to the Directory Server in lockdown mode receives an "Unavailable" response.

To Manually Enter Lockdown Mode

- Use `enter-lockdown-mode` to begin a lockdown mode.

```
$ bin/enter-lockdown-mode
```

To Leave Lockdown Mode

- Use `leave-lockdown-mode` to end lockdown mode.

```
$ bin/leave-lockdown-mode
```

To Start a Server in Lockdown Mode

- Use the --lockdownMode option with the `start-server` tool to start a server in lockdown mode.

```
$ bin/start-server --lockdownMode
```

Configuring Maximum Shutdown Time

During shutdown, some database checkpointing and cleaning threads may remain active even after the default time period on systems with very large or very busy database backends. If checkpointing or cleaning is aborted prematurely, it could possible lead to significantly longer startup times for the Directory Server. The Directory Server provides an option for administrators to set the maximum time a shutdown process should take. When a shutdown process is initiated, the server begins stopping all of its internal components and waits up to 5 minutes for all threads to complete before exiting.

Administrators can use the `dsconfig` tool to increase the maximum shutdown time to allow database operations to complete.

To Configure the Maximum Shutdown Time

- Use the `dsconfig` tool to increase the maximum shutdown time for your system. The following command increases the maximum shutdown time from 5 minutes to 6 minutes. The command allows time values of w (weeks), d (days), h (hours), m (minutes), s (seconds), ms (milliseconds).

```
$ bin/dsconfig set-global-configuration-prop --set "maximum-shutdown-time:6
 m"
```

Remember to include the LDAP or LDAPS connection parameters (e.g., host name, port, bindDN and bindDN password) with the `dsconfig` command.

The `maximum-shutdown-time` property can also be changed using the `dsconfig` tool in interactive mode. From the main menu, select **Global Configuration**, and then select the option to display advanced properties.

Working with Referrals

A referral is a redirection mechanism that tells client applications that a requested entry or set of entries is not present on the Directory Server but can be accessed on another server. Referrals can be used when entries are located on another server. The Directory Server implements two types of referrals depending on the requirement.

- **Referral on Update Plug-in**. The Directory Server provides a Referral on Update Plug-in to create any referrals for update requests (add, delete, modify, or modDN operations) on read-only servers. For example, given two replicated directory servers where one server is a master (read-write) and the other, a read-only server, you can configure a referral for any client update requests on the second directory server to point to the master server. If a client application sends an add request, for example, on the second directory server, the directory server responds with a referral that indicates any updates should be made on the master server. All read requests on the read-only server will be processed as normal.

- **Smart Referrals**. The Directory Server supports smart referrals that map an entry or a specific branch of a DIT to an LDAP URL. Any client requests (reads or writes) targeting at or below the branch of the DIT will send a referral to the server designated in the LDAP URL.

Specifying LDAP URLs

Referrals use LDAP URLs to redirect a client application's request to another server. LDAP URLs have a specific format, described in RFC 4516 and require that all special characters be properly escaped and any spaces indicated as "%20". LDAP URLs have the following syntax:

```
ldap[s]://hostname:port/base-dn?attributes?scope?filter
```

where

- *ldap[s]* indicates the type of LDAP connection to the Directory Server. If the Directory Server connects over a standard, non-encrypted connection, then ldap is used; if it connects over SSL, then ldaps is used. Note that any search request initiated by means of an LDAP URL is anonymous by default, unless an LDAP client provides authentication.
- *hostname* specifies the host name or IP address of the Directory Server.
- *port* specifies the port number of the Directory Server. If no port number is provided, the default LDAP port (389) or LDAPS port (636) is used.
- *base-dn* specifies the distinguished name (DN) of an entry in the DIT. The Directory Server uses the base DN as the starting point entry for its searches. If no base DN is provided, the search begins at the root of the DIT.
- *attributes* specifies those attributes for which the Directory Server should search and return. You can indicate more than one attribute by providing a comma-separated list of attributes. If no attributes are provided, the search returns all attributes.
- *scope* specifies the scope of the search, which could be one of the following: `base` (only search the specified base DN entry), `one` (only search one level below the specified base DN), `sub` (search the base entry and all entries below the specified base DN). If no scope is provided, the server performs a base search.
- *filter* specifies the search filter to apply to entries within the scope of the search. If no filter is provided, the server uses +.

Creating Referrals

You can create a smart referral by adding an entry with the referral and `extensibleObject` object classes or adding the object classes to a specific entry. The referral object class designates the entry as a referral object. The `extensibleObject` object class allows you to match the target entry by matching any schema attribute. The following example shows how to set up a smart referral if a portion of a DIT is located on another server.

To Create a Referral

1. Create an LDIF file with an entry that contains the `referral` and `extensibleObject` object classes.

```
dn: ou=EngineeringTeam1,ou=People,dc=example,dc=com
objectClass: top
objectClass: referral
objectClass: extensibleObject
ou: Engineering Team1
ref: ldap://server2.example.com:6389/
ou=EngineeringTeam1,ou=People,dc=example,dc=com
```

2. On the first server, add the referral entry using the `ldapmodify` command.

```
$ bin/ldapmodify --defaultAdd --fileName referral-entry.ldif
```

3. Verify the addition by searching for a user.

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com "(uid=user.4)"
```

```
SearchReference(referralURLs={ldap://server2.example.com:6389/
 ou=EngineeringTeam1,ou=People,dc=example,dc=com??sub?})
```

To Modify a Referral

- Use `ldapmodify` with the `manageDSAIT` control to modify the `ref` attribute on the referral entry.

```
$ bin/ldapmodify --control manageDSAIT
dn: ou=EngineeringTeam1,ou=People,dc=example,dc=com
changetype: modify
replace: ref
ref: ldap://server3.example.com/
ou=EngineeringTeam1,ou=People,dc=example,dc=com
```

To Delete a Referral

- Use `ldapdelete` with the `manageDSAIT` control to delete the referral entry.

```
$ bin/ldapdelete \
  --control manageDSAIT "ou=EngineeringTeam1,ou=People,dc=example,dc=com"
```

Configuring a Read-Only Server

The PingDirectory Server provides a means to configure a hub-like, read-only directory server for legacy systems that require it. The read-only directory server participates in replication but cannot respond to any update requests from an external client. You can configure the Directory Server by setting the writability mode to internal-only, which makes the server operate in read-only mode. Read-only mode directory servers can process update operations from internal operations but reject any write requests from external clients. Because the Directory Server cannot accept write requests, you can configure the server to send a referral, which redirects a client's request to a master server. The client must perform the operation again on the server named in the referral.

Note: For Implementers of Third Party Extensions. Many Server SDK extensions use the `InternalConnection` interface to process operations in the server, rather than issuing LDAP requests over the network. If an extension does so in response to an external update request, then any Directory Server using that extension will effectively respond to external update requests, even though the Directory Server is configured to operate in read-only mode, as described above. One possible workaround is to split the extension into two extensions, one for reads and one for writes, then disabling (or not deploying) the write-only extension when configuring a Directory Server in read-only mode.

To Configure a Read-Only Server

1. Install two replicating directory servers. See *Replication Configuration* for various ways to set up your servers.
2. On the second server, use the `dsconfig` command to set the writability mode of the server to internal-only.

```
$ bin/dsconfig set-global-configuration-prop \
  --set writability-mode:internal-only
```

3. On the second server, use the `dsconfig` tool to create a referral that instructs the server to redirect client write requests under `dc=example,dc=com` to server1.example.com:1389. The referral itself is defined as a plugin of type `Referral on Update`. This command sets up the server to process read operations but redirects all write operations under `dc=example,dc=com` to another server.

```
$ bin/dsconfig create-plugin --plugin-name "Refer Updates" \
  --type referral-on-update \
  --set enabled:true \
  --set referral-base-url:ldap://server1.example.com:1389/ \
  --set "base-dn:dc=example,dc=com"
```

4. To test the referral, attempt to modify an entry and confirm that the server responds with the result code of 10. The resulting message is available in the server's access log.

```
$ bin/ldapmodify -p 2389 -D "cn=Directory Manager" -w password
dn: uid=user.12,ou=People,dc=example,dc=com
changetype:modify
replace:telephoneNumber
telephoneNumber: +1 408 555 1155
```

```
[06/Aug/2012:15:28:21.468 -0400] MODIFY
RESULT conn=86 op=1 msgID=1 requesterIP="127.0.0.1"
dn="uid=user.12,ou=People,dc=example,dc=com" resultCode=10
referralURLs="ldap://server1.example.com:1389/uid=user.12,
ou=People,dc=example,dc=com" etime=0.223
```

Configuring HTTP Access for the Directory Server

Although most clients communicate with the PingDirectory Server using LDAP, the server also provides support for an HTTP connection handler that uses Java servlets to serve content to clients over HTTP. Ping Identity offers an extension that uses this HTTP connection handler to add support for the System for Cross-domain Identity Management (SCIM) protocol. Third-party developers can also use the Server SDK to write extensions that leverage this HTTP support.

The following sections describe how to configure HTTP servlet extensions and how to configure an HTTP connection handler.

Configuring HTTP Servlet Extensions

To use the HTTP connection handler, you must first configure one or more servlet extensions. Servlet extensions are responsible for obtaining Java servlets (using the Java Servlet 3.0 specification as described in JSR 315) and registering them to be invoked using one or more context paths. If you plan to deploy the SCIM extension, then you should follow the instructions in Chapter 24, "*Managing the SCIM Servlet Extension*." For custom servlet extensions created using the Server SDK, the process varies based on whether you are using a Java-based or Groovy-scripted extension.

Web Application Servlet Extensions

A Web application may be deployed either as a WAR file that has been packaged according to the standard layout and containing a web.xml deployment descriptor, or from a directory containing the application's source components arranged in the standard layout.

When deploying a Web application from a directory, you may specify the location of the `web.xml` deployment descriptor if it is not in the standard location. You may also specify the directory used by the server for temporary files. At runtime the web application has access to the server classes.

Java-based Servlet Extensions

For Java-based extensions, first use the Server SDK to create and build the extension bundle as described in the Server SDK documentation. Then, install it using the `manage-extension` tool as follows:

```
$ bin/manage-extension --install/path/to/extension.zip
```

The Java-based extension may then be configured for use in the server using `dsconfig` or the Administrative Console. Create a new Third Party HTTP Servlet Extension, specifying the fully-qualified name for the `HTTPServletExtension` subclass in the `extension-class` property, and providing any appropriate arguments in the `extension-argument` property.

Note: Web Application and Servlet extensions run in a shared embedded web application server environment. Incompatiblities or conflicts may arise from use of different versions of commonly used jars or including frameworks such as loggers, Spring components, JAX-RS implementations or other resources which may require a dedicated Java Virtual Machine (JVM) environment. After introducing a custom extension, check the server error log for an indication that the extension loaded successfully. The error log may also contain debug information if the extension failed to load with an initialization exception or did not complete initialization.

Groovy-Scripted Extensions

For Groovy-scripted extensions, place the necessary Groovy scripts in the appropriate directory (based on the package for those scripts) below the `lib/groovy-scripted-extensions` directory. Then, create a new Groovy Scripted HTTP Servlet Extension, specifying the fully-qualified Groovy class name for the `script-class` property, and providing any appropriate arguments in the `script-argument` property.

Configuring HTTP Operation Loggers

Servlet extensions may write error log messages in the same way as any other kind of server component, but interaction with HTTP clients will not be recorded in the server access log. However, if a servlet extension performs internal operations to interact with data held in the directory server, then those operations may be captured in the access log. To capture information about communication with HTTP clients, you must configure one or more HTTP operations loggers.

By default, the server comes with a single HTTP operation logger implementation, which uses the standard W3C common log format. It records messages in a format like the following:

```
127.0.0.1 - - [01/Jan/2012:00:00:00 -0600]"GET/hello HTTP/1.1" 200 113
```

The log message contains the following elements:

- The IP address of the client that issued the request.
- The RFC 1413 (ident) identity of the client. Because the ident protocol is not typically provided by HTTP clients, the HTTP connection handler never requests this information. This identity will always be represented as a dash to indicate that information is not available.
- The authenticated identity determined for the request by HTTP authentication, or a dash to indicate that the request was not authenticated.
- The time that the request was received.
- The request issued by the client, including the HTTP method, path and optional query string, and the HTTP protocol version used.
- The integer representation of the HTTP status code for the response to the client.
- The number of bytes included in the body of the response to the client.

To configure an HTTP operation logger to use this common log format, create a new instance of a Common Log File HTTP Operation Log Publisher object, specifying the path and name for the active log file to be written and the rotation and retention policies that should be used to manage the log files. In general, properties for Common Log File HTTP Operation Log Publisher objects have the same meaning and use as they do for other kinds of loggers.

You can use the Server SDK to create custom Java-based or Groovy-scripted HTTP operation loggers using the Third Party HTTP Operation Log Publisher and Groovy Scripted HTTP Operation Log Publisher object types.

Example HTTP Log Publishers

When troubleshooting HTTP Connection Handler issues, administrators should first look at the logs to determine any potential problems. The following section shows some `dsconfig` commands and their corresponding log files.

Default Configuration Example. You can configure a default detailed HTTP Log Publisher with default log rotation and retention policies as follows:

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Detailed Access Logger" \
  --type detailed-http-operation \
  --set enabled:true \
  --set log-file:logs/http-detailed-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

The corresponding log file provides access information below. The log message contains the following elements:

The time that the request was received.

The request ID issued by the client, including the IP address, port, HTTP method, and URL.

The authorization type, request content type, and status code.

The response content length.

The redirect URI.

The response content type.

The HTTP log file is shown as follows:

```
[23/Feb/2012:01:19:45 -0600] RESULT requestID=4300604 from="10.5.1.10:53269"
method="GET" url="https://10.5.1.129:443/Gimel/Users/
uid=user.402914,ou=People,
dc=gimel" authorizationType="Basic" requestContentType="application/json"
statusCode=200 etime=4.145 responseContentLength=1530 redirectURI="https://
x2270-11.example.lab:443/Gimel/Users/uid=user.402914,ou=people,dc=gimel"
responseContentType="application/json"
[23/Feb/2012:01:19:45 -0600] RESULT requestID=4300605 from="10.5.1.10:53269"
method="PUT" url="https://10.5.1.129:443/Gimel/Users/
uid=user.207585,ou=people,
dc=gimel" authorizationType="Basic" requestContentType="application/json"
statusCode=200 etime=4.872 responseContentLength=1532 redirectURI="https://
x2270-11.example.lab:443/Gimel/Users/uid=user.207585,ou=people,dc=gimel"
responseContentType="application/json"
[23/Feb/2012:11:31:18 -0600] RESULT requestID=4309872 from="10.5.1.10:3
```

Configuration with Request/Response Header Names and Values. The following example adds request/response header names and values, including the "Content-Type" request header, which is normally suppressed by default.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "HTTP Detailed Access Logger" \
  --set log-request-headers:header-names-and-values \
  --remove suppressed-request-header-name:Content-Type \
  --set log-response-headers:header-names-and-values
```

The following is a log example of a query request by a SCIM Server using the property, `scim-query-rate`. The log message contains the basic log elements shown in the first example plus the following additional information:

The request header for the host, http method, content type, connection, user agent.

The response header for the access-control credentials.

```
[23/Oct/2012:11:39:41-0600] RESULT requestID=4665307 from="10.5.0.20:56044"
method="GET" url="https://10.5.1.129:443/Beth/Users?attributes=userName,title,
emails,urn:scim:schemas:extension:custom:1.0:descriptions,urn:scim:schemas:
extension:enterprise:1.0:manager,groups,urn:scim:schemas:extension:custom:1.0:
blob&filter=userName+eq+%22user.18935%22" requestHeader="Host:
 x2270-11.example.
lab:443" requestHeader="Accept: application/json" requestHeader="Content-Type:
application/json" requestHeader="Connection: keep-alive"
requestHeader="User-Agent: Wink Client v1.1.2" authorizationType="Basic"
requestContentType="application/json" statusCode=200 etime=140.384
responseContentLength=11778 responseHeader="Access-Control-Allow-Credentials:
true" responseContentType="application/json"
```

Another log example shows an example user creation event. The client is curl.

```
[23/Oct/2016:11:50:11-0600] RESULT requestID=4802791 from="10.8.1.229:52357"
method="POST" url="https://10.2.1.113:443/Aleph/Users/" requestHeader="Host:
 x2270-
11.example.lab" requestHeader="Expect: 100-continue" requestHeader="Accept:
 applica-
tion/xml" requestHeader="Content-Type: application/xml" requestHeader="User-
Agent:
curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4 OpenSSL/0.9.8r
 zlib/1.2.5"
authorizationType="Basic" requestContentType="application/xml" requestContent-
Length=1773 statusCode=201 etime=11.598 responseContentLength=1472 redirec-
tURI="https://x2270-11.example.lab:443/Aleph/Users/b2cef63c-5e46-11e1-974b-
60334b1a0d7a" responseContentType="application/xml"
```

The final example shows a user deletion request. The client is the Sync Server.

```
[23/Oct/2016:11:38:06-0600] RESULT requestID=4610261 from="10.5.1.114:34558"
method="DELETE" url="https://10.2.1.113:443/Aleph/Users/b8547525-24e0-41ae-
b66b-
0b441800de70" requestHeader="Host: x2270-11.example.lab:443"
 requestHeader="Accept:
application/json" requestHeader="Content-Type: application/json"
 requestHeader="Con-
nection: keep-alive" requestHeader="User-Agent: DataSync-6.0.0.0 (Build
20121022173845Z, Revision 11281)" authorizationType="Basic"
 requestContentType="appli-
cation/json" statusCode=200 etime=10.615 responseContentLength=0
```

Configuring HTTP Connection Handlers

HTTP connection handlers are responsible for managing the communication with HTTP clients and invoking servlets to process requests from those clients. They can also be used to host web applications on the server. Each HTTP connection handler must be configured with one or more HTTP servlet extensions and zero or more HTTP operation log publishers.

If the HTTP Connection Handler cannot be started (for example, if its associated HTTP Servlet Extension fails to initialize), then this will not prevent the entire Directory Server from starting. The Directory Server's `start-server` tool will output any errors to the error log. This allows the Directory Server to continue serving LDAP requests even with a bad servlet extension.

The configuration properties available for use with an HTTP connection handler include:

- **listen-address**. Specifies the address on which the connection handler will listen for requests from clients. If not specified, then requests will be accepted on all addresses bound to the system.
- **listen-port**. Specifies the port on which the connection handler will listen for requests from clients. Required.
- **use-ssl**. Indicates whether the connection handler will use SSL/TLS to secure communications with clients (whether it uses HTTPS rather than HTTP). If SSL is enabled, then `key-manager-provider` and `trust-manager-provider` values must also be specified.
- **http-servlet-extension**. Specifies the set of servlet extensions that will be enabled for use with the connection handler. You can have multiple HTTP connection handlers (listening on different address/port combinations) with identical or different sets of servlet extensions. At least one servlet extension must be configured.
- **http-operation-log-publisher**. Specifies the set of HTTP operation log publishers that should be used with the connection handler. By default, no HTTP operation log publishers will be used.
- **key-manager-provider**. Specifies the key manager provider that will be used to obtain the certificate presented to clients if SSL is enabled.
- **trust-manager-provider**. Specifies the trust manager provider that will be used to determine whether to accept any client certificates presented to the server.
- **num-request-handlers**. Specifies the number of threads that should be used to process requests from HTTP clients. These threads are separate from the worker threads used to process other kinds of requests. The default

value of zero means the number of threads will be automatically selected based on the number of CPUs available to the JVM.

- **web-application-extension**. Specifies the Web applications to be hosted by the server.

To Configure an HTTP Connection Handler

An HTTP connection handler has two dependent configuration objects: one or more HTTP servlet extensions and optionally, an HTTP log publisher. The HTTP servlet extension and log publisher must be configured prior to configuring the HTTP connection handler. The log publisher is optional but in most cases, you want to configure one or more logs to troubleshoot any issues with your HTTP connection.

1. The first step is to configure your HTTP servlet extensions. The following example uses the ExampleHTTPServletExtension in the Server SDK.

```
$ bin/dsconfig create-http-servlet-extension \
  --extension-name "Hello World Servlet" \
  --type third-party \
  --set "extension-
class:com.unboundid.directory.sdk.examples.ExampleHTTPServletExtension" \
  --set "extension-argument:path=/" \
  --set "extension-argument:name=example-servlet"
```

2. Next, configure one or more HTTP log publishers. The following example configures two log publishers: one for common access; the other, detailed access. Both log publishers use the default configuration settings for log rotation and retention.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Common Access Logger" \
  --type common-log-file-http-operation \
  --set enabled:true \
  --set log-file:logs/http-common-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"

$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Detailed Access Logger" \
  --type detailed-http-operation \
  --set enabled:true \
  --set log-file:logs/http-detailed-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

3. Configure the HTTP connection handler by specifying the HTTP servlet extension and log publishers. Note that some configuration properties can be later updated on the fly while others, like `listen-port`, require that the HTTP Connection Handler be disabled, then re-enabled for the change to take effect.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "Hello World HTTP Connection Handler" \
  --type http \
  --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set "http-servlet-extension:Hello World Servlet" \
  --set "http-operation-log-publisher:HTTP Common Access Logger" \
  --set "http-operation-log-publisher:HTTP Detailed Access Logger" \
  --set "key-manager-provider:JKS" \
  --set "trust-manager-provider:JKS"
```

4. By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to TRUE by default. You can monitor the connection handler using the `ldapsearch` tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

To Configure an HTTP Connection Handler for Web Applications

1. Create the Web application servlet extension.

```
$ bin/dsconfig create-web-application-extension \
  --extension-name "Hello Web Application" \
  --set "base-context-path:/hello-app" \
  --set "document-root-directory:/opt/hello-web-app"
```

2. By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to TRUE by default. You can monitor the connection handler using the `ldapsearch` tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

Domain Name Service (DNS) Caching

If needed, two global configuration properties can be used to control the caching of hostname-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system. Use the `dsconfig` tool to configure these properties.

- **network-address-cache-ttl** – Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a hostname-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.
- **network-address-outage-cache-enabled** – Caches hostname-to-IP address results in the event of a DNS outage. This is set to true by default, meaning name resolution results are cached. Unexpected service interruptions may occur during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

IP Address Reverse Name Lookups

PingDirectory Server does not explicitly perform numeric IP address-to-hostname lookups. However, address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing may trigger implicit reverse name lookups. For more information about how address masks are configured in the server, review the following information for each server:

- ACI dns: bind rules under *Managing Access Control* (Directory Server and Directory Proxy Server)
- `ds-auth-allowed-address`: *Adding Operational Attributes that Restrict Authentication* (Directory Server)
- Connection Criteria: *Restricting Server Access Based on Client IP Address* (Directory Server and Directory Proxy Server)
- Connection Handlers: restrict server access using Connection Handlers (Configuration Reference Guide for all servers)

Configuring Traffic Through a Load Balancer

If a PingDirectory Server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, then it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request. If the actual client's IP address must be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and PingDirectory Server. For PingDirectory Servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP), and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

On the load balancer, configure settings to provide the `X-Forwarded-*` information, such as `X-Forwarded-Host:`. See the product documentation for the device type.

Working with the Referential Integrity Plug-in

Referential integrity is a plug-in mechanism that maintains the DN references between an entry and a group member attribute. For example, if you have a group entry consisting of member attributes specifying the DNs of printers, you can enable the referential integrity plug-in to ensure that the group entry is automatically removed if a printer entry is removed from the Directory Server.

The Referential Integrity Plug-in is disabled by default. When enabled, the plug-in performs integrity updates on the specified attributes (for example, member or uniquemember) after a delete, modify DN, or a rename (i.e., subordinate modifyDN) operation is logged to the `logs/ referint` file. If an entry is deleted, the plug-in checks the log file and makes the corresponding change to the associated group entry.

Three important points about the Referential Integrity Plug-in:

- All specified attributes that are configured for Referential Integrity must be indexed.
- On replicated servers, the Referential Integrity Plug-in configuration is not propagated to other replicas; therefore, you must manually enable the plug-in on each replica.
- The plug-in settings must also be identical on all machines.
- Subtree delete operations are not allowed if the referential integrity plugin is enabled and configured to operate in synchronous mode. It must be configured to operate in asynchronous mode (by specifying a nonzero update interval) if subtree delete operations will be performed.

To Enable the Referential Integrity Plug-in

1. Determine the attributes needed for your system. By default, the `member` and the `uniquemember` attributes are set for the plug-in.
2. Run the `dsconfig` tool to enable the Referential Integrity Plug-in.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

Working with the Unique Attribute Plug-in

The Unique Attribute plug-in is used to enforce uniqueness constraints on the values of one or more attributes across a portion of the Directory Server. The plug-in checks for uniqueness prior to an add, modify, or modify DN request and will instruct the server to reject the request if a constraint violation is found.

The plug-in is disabled by default as it can affect performance in heavy write load environments. Once the plug-in is enabled, it does not check for attribute uniqueness on existing entries, but only on new ADD, MODIFY, or MODDN operations. However, administrators can use the `identify-unique-attribute-conflicts` tool to ensure that no such conflicts exist in the data.

Important: All attributes for which uniqueness should be enforced should be indexed for equality in all backends. The LDAP SDK uniqueness request control can be used for enforcing uniqueness on a per-request basis. See the LDAP SDK documentation and the `com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl` class for using the control. See the ASN.1 specification to implement support for it in other APIs.

Attribute uniqueness can be enforced in replicated environments in which each replica contains the complete set of data for which to provide uniqueness, regardless of whether clients communicate directly with the server or interact with it through a Directory Proxy Server. In such environments, all servers should have identical uniqueness configurations. Note that it is not possible to *completely* prevent conflicts that arise from simultaneous writes on separate replicas. However, such conflicts will be detected after the changes have been replicated and will trigger administrative alert notifications.

For proxied environments that do not have the complete set of data on all servers (e.g., environments that use entry balancing or that store different portions of the DIT on different servers), you can implement the Global Uniqueness Attribute Plug-in on the Directory Proxy Server, instead of enabling the attribute uniqueness plug-in on the Directory Server. For more information, see the PingDirectoryProxy Server Administration Guide.

To Enable the Unique Attribute Plug-in

1. Determine which attributes must be unique in your data.
2. Run the `dsconfig` tool to enable the plug-in. By default, the plug-in type property is set to `postsynchronizationadd`, `postsynchronizationmodify`, `postsynchronizationmodifydn`, `preoperationadd`, `preoperationmodify`, and `preopertionmodifydn`. If you want to set one plug-in type, use the --set plugin-type:<operation-type> option. For example, use --set plugin-type:preoperationadd with the following command if you only want to check for attribute uniqueness prior to ADD operation.

```
$ bin/dsconfig set-plugin-prop --plugin-name "UID Unique Attribute" \
  --set enabled:true
```

Working with the Purge Expired Data Plug-in

The Purge Expired Data plug-in is used to delete expired entries or attribute values, and cleanup expired PingFederate Persistent Access Grants. When the plug-in is enabled, a background thread in the plugin periodically searches for and purges expired data. It is recommended that the Purge Expired Data plug-in be enabled on multiple servers in a topology. One server can be configured to delete data, while others can be searching for expired data.

The Purge Expired Data plug-in is created and configured with the `dsconfig` tool. Configuration options include the base DN and filter, the items to be purged, how to identify expired data, and the frequency for polling and purging. The search for expired data must be indexed. An alarm is raised if the server purging data falls behind the confiured `max-updates-per-second`. Monitoring information is available in the Admin Console, or `cn=monitor`.

To Configure the Purge Expired Data Plug-in for Expired Entries

This example deletes all unverified account entries that have not been accessed in the past eight weeks. They could be accounts that potential customers started to create through an application's registration process, but then did not complete. The phone number or email address that was provided during registration was not verified, and should be allowed to be used by another account. The server can track the last access time automatically in the `ds-last-access-time` attribute by enabling the Last Access Time plug-in.

1. If necessary, enable the Last Access Time plug-in:

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Last Access Time" \
  --set enabled:true
```

2. The Purge Expired Data Plug-in requires the date attribute that is used to determine expiration to be indexed for ordering. An index on that attribute must be created:

```
$ bin/dsconfig create-local-db-index \
  --backend-name userRoot \
  --index-name ds-last-access-time \
  --set index-type:ordering
```

3. If there is data present in the directory, rebuild the index:

```
$ bin/rebuild-index \
  --baseDN dc=example,dc=com \
  --index ds-last-access-time
```

4. Create the plug-in that purges account entries `objectclass=account` that are not verified `verified=false` after eight weeks of inactivity:

```
$ bin/dsconfig create-plugin \
  --plugin-name "Purge Old Unvalidated Accounts" \
  --type purge-expired-data \
  --set enabled:true \
  --set datetime-attribute:ds-last-access-time \
  --set "expiration-offset:8 w" \
  --set "filter:(&(objectClass=account)(verified=false))"
```

To Configure the Purge Expired Data Plug-in for Expired Attribute Values

The Purge Expired Data plug-in can also be used to delete values of an attribute that have expired. For example, an application may track information about an employee's session, but the session should expire after 24 hours. There may be multiple active sessions tracked across different devices, with session information that looks like this:

```
sessionInfo: { “sessionId” : “E85FAC04E331FFCA55549B10B7C7A4FA”,
"ipAddress": "10.0.0.00", "userAgent": "Mozilla/5.0 (iPad; U; CPU OS 3_2
 like Mac OS X; en-us)
AppleWebKit/531.21.10 (KHTML, like Gecko) Version/4.0.4 Mobile/7B367
 Safari/531.21.10",
“creationTime” : “2018-03-31T13:10:15Z” }
```

In this example, the LDAP attribute is `sessioninfo`, and the JSON field that stores the time stamp is `creationTime`. These will be used to configure the Purge Expired Data plug-in.

To purge the JSON attribute values after 24 hours, rather than the entire session entry, the plug-in can be created with the following steps.

1. Create an index on the `creationTime` field of the `sessioninfo` attribute:

```
$ bin/dsconfig create-json-attribute-constraints \
  --attribute-type sessioninfo \
  --set enabled:true
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type sessioninfo \
  --json-field creationTime \
  --set index-values:true \
  --set value-type:string
```

2. Create and enable the plug-in:

```
$ bin/dsconfig create-plugin \
  --plugin-name "Purge Old Session Data Plugin" \
  --type purge-expired-data \
  --set enabled:true \
  --set "custom-datetime-format:yyyy-MM-dd'T'HH:mm:ss'Z'" \
  --set datetime-attribute:sessioninfo \
  --set datetime-format:custom \
  --set datetime-json-field:creationTime \
  --set "expiration-offset:1 d" \
  --set purge-behavior:delete-json-attribute-values
```

Configuring Uniqueness Across Attribute Sets

Attribute uniqueness can be configured across a set of attributes using the `multiple-attribute-behavior` property. The `multiple-attribute-behavior` property can take the following values:

- **unique-within-each-attribute** - If multiple attributes are specified, then uniqueness will be enforced for all values of each attribute, but the same value may appear in different attributes (in the same entry or in different entries). For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-within-each-attribute"` and add:

  An entry with a `telephoneNumber` value that matches the `telephoneNumber` attribute in the existing entry, then the add request will fail.

  An entry with a `mobile` value that matches the `mobile` attribute in the existing entry, then that too will fail.

  An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., 123-456-7893) but differ from the values in the existing entry, then the add request will succeed.

- **unique-across-all-attributes-including-in-same-entry** - If multiple attributes are specified, then uniqueness will be enforced across all of those attributes, so that if a value appears in one of those attributes, that value may not be present in any other of the listed attributes in the same entry, nor in any of the listed attributes in other entries. For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-across-all-attributes-including-in-same-entry"` and add:

  An entry with a `telephoneNumber` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then the add request will fail.

  An entry with a `mobile` value that matches the `mobile` attribute in an existing entry, then that too will fail.

  An entry with a `mobile` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then that will fail.

  An entry with a `telephoneNumber` value (e.g., 123-456-7891) that matches the `mobile` attribute in an existing entry, then that too will fail.

  An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., 123-456-7893) but differ from the values in an existing entry, then the add request will fail.

- **unique-across-all-attributes-except-in-same-entry** - If multiple attributes are specified, then uniqueness will be enforced across all of those attributes, so that if a value appears in one of those attributes, that value may not be present in any of the listed attributes in other entries. However, the same value may appear in multiple attributes in the same entry. For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-across-all-attributes-except-in-same-entry"` and add:

  An entry with a `telephoneNumber` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then the add request will fail.

  An entry with a `mobile` value that matches the `mobile` attribute in the existing entry, then that too will fail.

  An entry with a `mobile` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then that will fail.

  An entry with a `telephoneNumber` value (e.g., 123-456-7891) that matches the `mobile` attribute in an existing entry, then that will fail.

  An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., 123-456-7893) but differ from the values in an existing entry, then the add request will succeed.

To Enable Uniqueness Across Attribute Sets

- Use `dsconfig` to configure the UID Unique Attribute plug-in to apply across multiple attributes. The `multiple-attribute-behavior` property is set to "`unique-within-each-attribute`", which indicates that uniqueness will be enforced for all values of each attribute (e.g., telephoneNumber=123-456-7890 and mobile=123-456-7891), but the same value (e.g., either 123-456-7890 or 123-456-7891) may appear in different attributes in the same entry or in different entries.

```
$ dsconfig create-plugin \
  --plug-in "Unique telephoneNumber and mobile" \
  --type "unique-attribute" \
  --set "enabled:true" \
  --set "type:telephoneNumber" \
  --set "type:mobile" \
  --set "multiple-attribute-behavior:unique-within-each-attribute" \
  --no-prompt
```

Working with the Last Access Time Plug-In

The Last Access Time plug-in is used to record the timestamp of the last activity targeting an entry. The plug-in updates the `ds-last-access-time` attribute of the entry when it is accessed by an add, bind, compare, modify, modify DN, or search operation.

The plug-in can be used with the Directory Server Uncached Attribute Criteria, or any application that needs to determine how recently an entry has been accessed. The plug-in also enables defining request criteria to limit the scope of tracking the last access time. The `max-search-result-entries-to-update` property also prevents mass updates of `ds-last-access-time` when searches contain many results, but may not reflect end-user access. Consider the following when using this plug-in:

- The plugin should be enabled on all servers that have the same configuration.
- An updated `ds-last-access-time` attribute value is replicated like any other change to an entry.
- The `ds-last-access-time` attribute is not returned from a search, unless included in the attributes list explicitly, or given the "+" specification for operational attributes.
- The `ds-last-access-time` value format is `yyyyMMddHHmmss.SSS'Z'`, which provides millisecond-level accuracy, such as `20131207144135.821Z`.
- The `ds-last-access-time` attribute can be indexed with a local database index. The ordering index type is the most relevant, but may require a higher index entry limit (default is 4000) to accommodate searches for entries that have not been accessed in a long period of time. The ordering index type, with a short time range or high index entry limit, will result in indexed search results for requests such as `(ds-last-access-time>=20131207144135.821Z).`

Important: Deployments prior to version 4.5 using the last access time plug-in should disable the plug-in before upgrading, and then re-enable the plug-in once the update is complete. If servers are running different versions, the `last-access-time` updates may occur with a higher frequency than intended.

Working with the Pass Through Authentication Plug-In

The Pass Through Authentication plugin is used to delegate bind operations to remote LDAP servers by forwarding simple bind requests to an external LDAP server, including Active Directory. The plugin can be configured to attempt a local bind, set or update a local password, and bypass local password policies to ensure remote passwords are migrated.

Consider the following when using this plugin:

- The plugin should be enabled on all servers that have the same configuration.
- Remote servers accepting a forwarded bind request may require connection security, such as a secure StartTLS or LDAPS TLS connection.
- Carefully consider how password changes and password resets are handled. Updating a password in the Directory Server may result in divergent passwords between the local and remote server. The Data Sync Server can be used to synchronize passwords between servers, if needed.
- The plugin only updates local passwords if the forwarded simple bind is successful. Expired passwords on a remote server may return an invalid credentials error causing the overall bind operation to fail.

- Multiple remote servers can be specified. The server-access-mode property determines if the servers are accessed in round-robin, failover-on-unavailable, or failover-on-any-failure modes. The default server access mode is round-robin.
- The update-local-password property indicates whether the local password value should be updated to the value used in the bind request, in the event that the local bind fails but the forwarded bind succeeds. A local entry must previously exist in order to update passwords.
- The allow-lax-pass-through-authentication-passwords property indicates whether updates to the local password value should accept passwords that do not meet local password policy requirements.
- The connection-criteria property specifies a set of connection criteria that must match the client associated with the local bind request for the bind to be forwarded to the remote server.
- The request-criteria property specifies a set of request criteria that must match the local bind request or a local target entry for the bind to be forwarded to the remote server.
- The dn-map property specifies one or more DN mappings that can be used to transform bind DNs before attempting to forward the bind to remote servers.
- The search-base-dn property is used when searching for a remote user entry using a filter constructed from the pattern defined in the search-filter-pattern property. It is not possible to configure both a DN map and search filter pattern. If neither a DN map nor a search filter pattern is defined, then user entries are expected to have the same DN in the local server and the remote servers.

Supporting Unindexed Search Requests

By default, the Directory Server denies all unindexed search requests, except for those issued by the bind DNs that have the unindexed-search privilege. This default behavior keeps the server from tying up worker threads on time-consuming, unindexed searches. However, you can turn off the enforcement of the unindexed-search privilege to allow any client to perform an unindexed search. To do this, set the disabled-privilege global configuration property to unindexed-search as follows:

```
$ bin/dsconfig set-global-configuration-prop \
  --set disabled-privilege:unindexed-search
```

If you choose to allow unindexed searches, you may want to cap the maximum number of concurrent unindexed search requests using the maximum-concurrent-unindexed-searches global configuration property. You configure this property using dsconfig as follows:

```
$ bin/dsconfig set-global-configuration-prop \
  --set maximum-concurrent-unindexed-searches:2
```

You can limit unindexed search privileges for particular clients using the allow-unindexedsearches property of the Client Connection Policy. For more information about configuring Client Connection Policies, see “Configuring Client Connection Policies”.

Sun/Oracle Compatibility

For companies that are migrating from a Sun/Oracle server to the PingDirectory Server, the PingDirectory Server provides a dsconfig batch file, sun-ds-compatibility.dsconfig, which describes the various components that can be configured to make the server exhibit behavior closer to a Sun/Oracle configuration.

Administrators can use the sun-ds-compatibility.dsconfig batch file to apply the Directory Server’s configuration with the necessary dsconfig commands. Simply uncomment the example commands listed in the file, and then run the dsconfig command specifying the batch file. Note that this batch file is not comprehensive and must be used together with the migrate-sun-ds-config tool, located in the bin folder (or bat folder for Windows systems) during the migration process. Both the tool and the batch file overlap in some areas but provide good initial migration support from the Sun/Oracle server to a Ping Identity server.

Another useful tool is the migrate-ldap-schema tool in the bin folder (or bat folder for Windows systems), which migrates schema information from an existing LDAP server onto this instance. All attribute type and object class definitions that are contained in the source LDAP server will be added to the targeted instance or written to a schema file.

To Configure the Directory Server for Sun/Oracle Compatibility

1. From the Directory Server server root directory, open the sun-ds-compatibility.dsconfig file in the docs folder. You can use a text editor to view the file.
2. Read the file completely.
3. Apply any changes to the file by removing the comment symbol at any dsconfig command example, and then applying the dsconfig command specifying the batch file.

```
$ bin/dsconfig --no-prompt --bindDN "cn=Directory Manager" \
  --bindPassword "password" --batch-file /path/to/dsconfig/file
```

4. Run the migrate-ldap-schema tool to move the schema definitions on the source server to the destination Ping Identity server.

```
$ bin/migrate-ldap-schema
```

5. Next, run the migrate-sun-ds-config tool to see what differences exist in the Ping Identity configuration versus the Sun/Oracle configuration. On the PingDirectory Server, run the following command:

```
$ bin/migrate-sun-ds-config
```

6. Test the server instance for further settings that may not have been set with the batch file, the migrate-ldap-schema tool, or the migrate-sun-ds-config tool.
7. If you notice continued variances in your configuration, contact your authorized support provider.

Chapter

7

Configuring Soft Deletes

Topics:

The PingDirectory Server (version 3.2.4 or later) supports a *soft-delete* feature that preserves a deleted entry's attribute and uniqueness characteristics to allow it to be undeleted or permanently removed at a later date.

This chapter introduces the following topics:

About Soft Deletes

The standard implementation of an LDAP server allows for adding, renaming, modifying, searching, comparing, and deleting one or more entries. The DELETE operation is, by specification, a destructive operation that permanently removes an entry and its attributes in a Directory Information Tree (DIT) but records the changes in access, and optionally, audit and change logs. During the DELETE operation, any associations such as references and group memberships are severed to reflect the entry that is removed. Meta attributes like operational attributes, which may be unique to an entry like `entryUUID`, will be lost or be different if the same entry is re-added to the Directory Server.

There are cases, however, where a company may want to preserve their deleted entries to allow for possible undeletion at a later date. For example, a company may want to retain account and subscriber entries for their users (e.g., customers, employees, or partners) who leave but later rejoin. Artifacts that a user creates such as account histories, web pages, notes, may be tracked and recovered while a user is deleted or when the user returns as an active customer.

To facilitate this use case, the Directory Server supports a feature called *soft deletes*, which preserves a deleted entry's attributes and entry uniqueness characteristics to allow the entry to be undeleted or permanently removed at a later date. A delete request may result in a soft delete either by the client explicitly requesting a soft delete or by the request matching criteria defined in an active soft delete policy. The soft-deleted entries are renamed by prefixing an `entryUUID` operational attribute to the DN and adding an auxiliary object class, `ds-soft-delete-entry`, to the entry, which saves the entry in a hidden state. All active references and group memberships are then removed. Once in this hidden state, soft-deleted entries are inaccessible to clients under normal operating conditions. Only clients with the `soft-delete-read` privilege will be allowed to interact with soft-deleted entries.

To allow soft deletes, the Directory Server's attribute uniqueness function has been relaxed to allow for the co-existence of a soft-deleted entry and an active entry with identical naming attributes, such as `uid`. For example, if a user John Smith was soft deleted but a different John Smith was added to the user accounts system, both entries could reside in the DIT without conflict: one in a soft-deleted state; the other, in an active state. The Directory Server extends this capability further by allowing multiple users with the same DN, who would normally conflict if active, to reside in the soft-deleted state.

Soft-deleted entries can be restored with an Undelete operation. However, the same uniqueness constraints that apply when adding a new user to the Directory Server are enforced when a soft-deleted entry is undeleted. Returning to the previous example, John Smith was soft deleted but a different John Smith with the same `uid` as the original John Smith was added later to the system. If the original John Smith was undeleted from its soft-deleted state, it would result in a conflict with the active John Smith entry. Administrators will need to modify the DN of the soft-deleted entry to avoid such conflicts.

Administrators can permanently remove a soft-deleted entry by performing a regular DELETE operation on it. This operation, called a *hard delete*, permanently removes a soft-deleted entry from the server. Also, note that you can also permanently remove a regular non-soft-deleted entry using a hard delete. This is useful when the server is configured with a soft-delete policy that would otherwise turn a regular delete request into a soft delete.

The Directory Server provides tool arguments that can use the Soft Delete Request Control, a Hard Delete Request Control, and other controls necessary to process these operations. Procedures to show how to use these options are presented later in this section.

For replicated topologies, when a participating directory server soft deletes an entry, it notifies the other replicas in the topology to soft delete the same entry on its respective machine. The changelog backend also records these entries by annotating them using an attribute that indicates its soft-deleted state. Modification and hard deletes of soft-deleted entries are not recorded by default in the changelog but can be enabled in the server. For maximum compatibility, it is highly recommended that all servers in the replication cluster support Soft Deletes and have identical Soft Delete configurations.

General Tips on Soft Deletes

There are some general tips about soft deletes that administrators should be aware of:

- **LDAP SDK and Server SDK**. The LDAP SDK and Server SDK both fully supports soft-deletes.
- **Possible Performance Impact for Searching Regular Entries and Soft-Deleted Entries**. There is little performance difference between retrieving a regular entry and a soft-deleted entry, respectively. However, there may be a performance impact when a search operation has to match criteria (such as, `uid=john.smith`) for both active entries and soft-deleted entries. For example, if there is one active `uid=john.smith` entry and two soft-deleted `uid=john.smith` entries, it may take the server a little more time to retrieve and try to match the criteria before it can return the results.
- **Soft Delete for Uncached Attributes and Entries**. The soft delete feature fully supports uncached attributes and uncached entries. See the section on *Uncached Attributes and Entries* for more information about the feature.
- **Soft Delete for Leaf Nodes Only**. Soft-deletion of any parent entry is not allowed. Likewise, soft-deleted entries that have soft-deleted sub-entries are not allowed.
- **Attempting to Soft-Delete a Soft-Deleted Entry Fails**. There are two available state options for soft-deletes: administrators can permanently delete a soft-deleted entry or undelete the entry. An administrator cannot soft-delete an already soft-deleted entry, which returns an `UNWILLING_TO_PERFORM` result code.
- **Soft-Deleted Users Have No Privileges**. Soft-deleted users do not have the ability to bind to the Directory Server or have authentication access. They cannot change their passwords and cannot undelete themselves. Also, soft-deleted entries cannot be used as an authorization identity using the proxied authorization or immediate client control. It is important to note that the soft-delete process does not destroy privilege assignment. If a soft-deleted entry is undeleted, the restored entry will retain the same privileges it originally had before being soft deleted. (One possible exception to this are those privileges assigned by virtual attributes that no longer match the newly-undeleted entry; those entries do not retain their original privileges.)
- **Soft-Deleted Entries Not Accessible by Other Means**. Soft-deleted entries may not be accessible from alternate access methods like SCIM.
- **Soft-Deleted Entries Can Be Modified but Not Renamed**. Administrators can search for all soft-deleted entries and the original source entry attributes can be updated as long as the administrators has modify privileges and access to the `soft-delete-read` privilege. Any attempt to rename a soft-deleted entry using a MODIFY DN operation will result in an `UNWILLING_TO_PERFORM` result code.
- **Replication**. Replication will have access to the LDAP operations with Soft Delete controls. These operations are transmitted, processed, and replayed as high-level requests, which are re-played on remote replicated servers. The replication conflict-resolution mechanism handles soft-deleted entries like any regular entries. For example, if a soft delete is executed independently on two servers then replicated, this results in a replication conflict. For maximum compatibility, it is highly recommended that all servers in a replication cluster support Soft Deletes and have identical Soft Delete configuration.
- **Transactions**. Soft-deletes are supported in transactions. The processing workflow uses the transactions mechanism and maintains the context information necessary to rollback failures to soft delete or undelete.
- **Soft-Deletes Through the Directory Proxy Server**. There is no special configuration steps to configure Soft Deletes on the Directory Proxy Server. The soft-deleted entry is routed directly to the underlying Directory Server. There is one exception: in an entry-balancing deployment, the Directory Proxy Server is responsible for routing the soft-deleted entry to the Directory Server containing the originally soft-deleted item. Also, as with standard entry-balanced deployments, it is not possible to undelete (using MODDN) an entry to a different Directory Server.
- **Export-LDIF**. The default behavior is to include soft-deleted entries as part of the `export-ldif` operation. If soft-deleted entries are to be excluded from export, administrators can use the --excludeSoftDeleteEntries option to filter out the entries.
- **Proxied Authorization**. The Soft Delete feature can be used with users who have proxied authorization privileges.
- **Ignored by Data Sync Server Sync Pipes**. For customers using the PingDataSync Server, soft-deleted entries are not synchronized by the server. Modifications or deletes of a soft-deleted entry are ignored by the Data Sync Server, and do not appear in the changelog by default. An actual soft delete operation appears to the changelog as a regular DELETE, and an actual undelete operation appears in the changelog as a regular ADD.
- **Referential Integrity Plugin Does Not Restore Membership**. References to a deleted DN are not restored by the referential integrity plugin upon undeletion of a soft-deleted entry. For example, if you have referential integrity enabled and you soft-delete a DN that is a member of a static group, the referential integrity plugin will remove

this DN from the group's list of members. When you undelete the soft-deleted entry, the plugin will not add the entry back to the group.

- **Criteria-Selected or Explicitly-Requested Purging of Soft Deletes**. The Soft Delete Policy configuration supports two new properties, `soft-delete-retention-time` and `soft-delete-retain-number-of-entries` that performs purging of soft deleted entries. See the section on *Configuring Soft-Delete Automatic Purging*.
- **Assigning Access to Controls to Non-Root Users Administrators**. By default, the root user account (e.g., `cn=Directory Manager`) has access to all of the controls needed to run the Soft Delete operations. For non-root users, you must grant access to these Soft Delete controls using access control rules. An example is shown in step 1 of the section, *To Configure Soft Deletes as a Global Configuration*. The following Soft Delete Controls are available to non-root users:

  Soft Delete Request Control. Allows the user to perform a soft delete operation. The OID for the control is 1.3.6.1.4.1.30221.2.5.20.

  Soft Delete Response Control. Allows the server to hold the DN of the soft-deleted entry that results from a soft delete request. The OID for the control is 1.3.6.1.4.1.30221.2.5.21.

  Hard Delete Request Control. Allows the user to run a hard delete operation on the entry, regardless if it is a regular or soft-deleted entry. The OID for the control is 1.3.6.1.4.1.30221.2.5.22.

  Undelete Request Control. Allows the user to undelete a soft-deleted entry using an ADD request. The OID for this control is 1.3.6.1.4.1.30221.2.5.23.

  Soft-Deleted Entry Access Request Control. Allows the user to search for any soft-deleted entries. The OID for this control is 1.3.6.1.4.1.30221.2.5.24. Note that a bind DN with the `stream-values` privilege can perform operations that can reveal soft-deleted entries, even if that bind DN does not have permission to use the Soft-Deleted Entry Access Request Control. For example, if a user can successfully run `dump-dns` or `ldap-diff`, then that user can get a list of soft-deleted entry DNs or soft-deleted entry contents via the output of one of those tools.

Configuring Soft Deletes on the Server

Soft deletes are configured on the Directory Server in several ways:

- **Using a Soft-Delete Policy and Global Configuration Property**. You can configure soft deletes by creating a soft-delete policy and a global configuration property. The soft-delete policy enables the feature on the server, while the global configuration property sets the controls used for the soft-delete requests. To enter a soft delete request, the `ldapmodify` or `ldapdelete` command requires the --useSoftDelete option. A delete request that does not have the --useSoftDelete option is treated as a hard delete, which permanently removes the entry.
- **Using Connection Criteria**. You can configure soft deletes by defining connection criteria within a client connection policy. Any clients that meet the criteria will have their deletes processed as soft deletes.
- **Using Request Criteria**. You can configure soft deletes by defining request criteria within a client connection policy. Any client requests that meet the criteria will have their deletes processed as soft deletes. Note that you can define both connection criteria and request criteria. Both criteria are exclusive and can exist within a soft-delete policy. In this case, the connection criteria is processed first, then the request criteria.

Configuring Soft Deletes as a Global Configuration

You can configure the soft delete feature by creating a soft-delete policy and then setting the configuration property on the server. The presence of the soft-delete policy enables the feature on the server and allows the global configuration property to send the necessary soft-delete requests.

This configuration setting requires that the --useSoftDelete option be used together with the `ldapmodify` or `ldapdelete` commands to send the delete using the Soft Delete Request Control. Without the --useSoftDelete option, any delete will be processed as a hard delete.

To Configure Soft Deletes as a Global Configuration

1. Create a non-root user admin account, such as `uid=admin,dc=example,dc=com`. See steps 1 and 3 in the Configuring Administrator Accounts section for more information. The following LDIF adds ACI's to the

uid=admin,dc=example,dc=com entry to allow access to user and operational attributes as well as access to the controls necessary to carry out the Soft Delete operations. The LDIF also adds the soft-delete-read privilege, which allows the user to access soft-deleted entries. Add the LDIF file using the ldapmodify tool.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*||+")
      (version 3.0; acl "Allow admins to read and write all user and
 operational
       attributes";
       allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.21")
      (version 3.0; acl "Allow admins to use the Soft Delete Request Control
 and
       Soft Delete Response Control";
       allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.22")
      (version 3.0; acl "Allow admins to use the Hard Delete Request
 Control";
       allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.23")
      (version 3.0; acl "Allow admins to use the Undelete Request Control";

       allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.24")
      (version 3.0; acl "Allow admins to use the Soft-Deleted Entry Access
 Request
       Control";
       allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)

$ bin/ldapmodify --fileName add-admin-aci.ldif
```

2. Create another LDIF file and assign the soft-delete-read privilege to the authorized user. Add the file to the server using the ldapmodify tool. You'll need to include bind credentials with the ldapmodify command.

```
dn: uid=admin,dc=example,dc=com
changeType: modify
add: ds-privilege-name
ds-privilege-name: soft-delete-read

$ bin/ldapmodify --fileName add-admin-priv.ldif
```

3. Configure a soft delete policy using the dsconfig command. The soft delete configuration option requires a soft-delete policy, which effectively enables the feature on the server. This is a required step.

```
$ bin/dsconfig create-soft-delete-policy \
  --policy-name default-soft-delete-policy
```

4. Configure the soft delete as a global configuration property using the dsconfig command. The command sets up the soft-delete controls necessary to send them as a request. This is a required step.

```
$ bin/dsconfig set-global-configuration-prop \
  --set soft-delete-policy:default-soft-delete-policy
```

After a successful modification, the server issues a warning that soft deletes are not enabled and that administrative action may be needed to prune older soft-deleted entries if resources are limited. For more information on setting up soft deletes, see “Configuring Soft Deletes by Connection Criteria”.

```
One or more configuration property changes require administrative action or
confirmation/notification. Those properties include:

 * soft-delete-policy:  Presently the server does not have an automatic
 purging
   capability for soft deleted entries. The administrator must
   periodically monitor and determine if too many soft deleted entries are
   consuming resources and must be pruned.

The Global Configuration was modified successfully
```

5. Delete an entry to test the soft delete feature. You must use the --useSoftDelete (or the short form, -s) option with the `ldapdelete` or `ldapmodify` to send the delete using a Soft Delete Request Control. The control renames the entry, prefixes an `entryUUID` operational attribute to the DN, and hides the entry by adding an auxiliary object class, `ds-soft-delete-entry`. If you do not include the --useSoftDelete option, the server processes the delete as a hard delete, which permanently removes the entry unless soft deletes is enabled. Upon a successful soft delete operation, the server returns the DN of the newly soft-deleted entry.

```
$ bin/ldapdelete --useSoftDelete uid=user.1,ou=People,dc=example,dc=com
```

```
Processing DELETE request for uid=user.1,ou=People,dc=example,dc=com
DELETE operation successful for DN uid=user.1,ou=People,dc=example,dc=com

Soft-deleted entry DN:
entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4+uid=user.1,ou=People,dc=exam-
ple,dc=com
```

Note:

Once an entry has been soft deleted, it is possible for an LDAP client to create an entry with the same DN as the original user entry DN. For example, if you soft-delete an entry (e.g., `uid=user.3,ou=People,dc=example,dc=com`), you can create a new entry using the same DN as the original entry, `uid=user.3,ou=People,dc=example,dc=com`. If the soft-deleted entry is undeleted at a later date, then a conflict will occur.

Searching for Soft Deletes

Soft-deleted entries are excluded from normal LDAP searches because they represent "deleted" entries. The `ldapsearch` tool has been updated to support these types of searches. If you want the option to search for soft-deleted entries, there are three ways to do so:

- **Base-Level Search on a Soft-Deleted entry by DN**. Use `ldapsearch` and specify the base DN of the specific soft-deleted entry that you are searching for.
- **Filtered Search by `soft-delete-entry` object class**. To search for all soft-deleted entries, use `ldapsearch` with a filter on the `ds-soft-delete-entry` objectclass.
- **Soft-Delete-Entry-Access Control**. You can use the Soft Delete Entry Access Control with the LDAP search to return soft-deleted entries. The `ldapsearch` tool provides a shortcut option, `--includeSoftDeletedEntries`, that sends the control to the server for processing. The control allows for the following search possibilities:

  Return only soft-deleted entries.
  Return non-deleted entries along with soft-deleted entries.
  Return only soft-deleted entries in undeleted form.

To Run a Base-Level Search on a Soft-Deleted Entry

- Run `ldapsearch` using the base DN of the specified soft-deleted entry.

```
$ bin/ldapsearch \
  --baseDN entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com \
  --searchScope base "(objectClass=*)"
```

```
# Soft-deleted entry DN:
# entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-soft-delete-entry
postalAddress: Aartjan Aalders$59748 Willow Street$Green Bay, TN 66239
postalCode: 66239
description: This is the description for Aartjan Aalders.
uid: user.1
userPassword: {SSHA}RdBCwQ2kIw57LukRthjrFBS/oFylJARnmTnorA==
employeeNumber: 1
initials: AKA
givenName: Aartjan
pager: +1 197 025 3730
mobile: +1 890 430 9077
cn: Aartjan Aalders
sn: Aalders
telephoneNumber: +1 094 100 7524
street: 59748 Willow Street
homePhone: +1 332 432 4295
l: Green Bay
mail: user.3@maildomain.net
st: TN
```

To Run a Filtered Search by soft-delete-entry Object Class

- Run `ldapsearch` to retrieve all soft-deleted entries using the `ds-soft-delete-entry` object class. The following command retrieves all soft-deleted entries on the server

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  "(objectclass=ds-soft-delete-entry)"
```

To Run a Search using the Soft Delete Entry Access Control

The following examples use the --includeSoftDeleteEntries `{with-non-deleted-entries | without-non-deleted-entries | deleted-entries-in-undeleted-form}` option, which uses the Soft Delete Entry Access Control. You could also use the --control option with the Soft Delete Entry Access Control symbolic name, `softdeleteentryaccess`, or the `--control` option with the actual Soft Delete Entry Access Control OID, `1.3.6.1.4.1.30221.2.5.24`.

- **Return Only Soft-Deleted Entries**. Run `ldapsearch` using the --includeSoftDeletedEntries option with the value of `without-non-deleted-entries` to return only soft-deleted entries.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries without-non-deleted-entries \
  --searchScope sub "(objectclass=*)"
```

- **Return Non-Deleted Entries Along with Soft-Deleted Entries**. Run `ldapsearch` using the --includeSoftDeletedEntries option with the value of with-non-deleted-entries to return non-deleted entries along with soft-deleted entries.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries with-non-deleted-entries \
  --searchScope sub "(objectclass=*)"
```

- **Return Only Soft-Deleted Entries in Undeleted Form**. Run `ldapsearch` using the --includeSoftDeletedEntries option with the value of `deleted-entries-in-undeleted-form` to return only soft-deleted entries in undeleted form. Some applications require access to all entries in the server, including both active and soft-deleted entries. The following command returns all entries that were soft-deleted but presents it in a form that is similar to a regular entry with the soft-delete DN in comments. This regular entry format does not show the actual soft-deleted DN but displays it in an "undeleted" form even though it is not actually "undeleted". Also, the object class, `ds-soft-delete-entry`, is not displayed:.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries deleted-entries-in-undeleted-form \
  --searchScope sub "(ds-soft-delete-from-dn=*)"

# Soft-deleted entry DN:
# entryUUID=2b5511e2-7616-389b-ab0c-025c805ad32c
+uid=user.14,ou=People,dc=exam-
ple,dc=com
dn: uid=user.14,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Abdalla Abdou$78929 Hillcrest Street$Elmira, ME 93080
postalCode: 93080
description: This is the description for Abdalla Abdou.
uid: user.14
userPassword: {SSHA}7GkzWiMiU12m5m+xBV+ZsoX3gVacMcRtSwDTFg==
employeeNumber: 14
initials: AFA
givenName: Abdalla
pager: +1 307 591 4870
mobile: +1 401 069 1289
cn: Abdalla Abdou
sn: Abdou
telephoneNumber: +1 030 505 6190
street: 78929 Hillcrest Street
homePhone: +1 119 487 2328
l: Elmira
mail: user.14@maildomain.net
st: ME
```

Undeleting a Soft-Deleted Entry Using the Same RDN

To undelete a soft-deleted entry, use `ldapmodify` with the --allowUndelete option and target the specific soft-deleted entry that you want to restore. In an LDIF file or from the command line, specify the `dn:<target entry>` attribute, which is the DN that the entry will be undeleted *to* and the `ds-undelete-from-dn` attribute, which is the entry that will be undeleted *from*. An undelete requires the `add` changetype so that the entry can be re-added to the server.

To Undelete a Soft-Deleted Entry Using the Same RDN

- Use `ldapmodify` with the --allowUndelete option and target the specific soft-deleted entry that you want to restore. The --allowUndelete option sends the Soft Undelete Request Control to the server. The first DN is the entry to undelete to and the `ds-undelete-from-dn` is the soft-delete entry to undelete from.

```
$ bin/ldapmodify --allowUndelete
dn: uid=user.1,ou=People,dc=example,dc=com
changetype:add
ds-undelete-from-dn: entryUUID=4e9b7847-edcb-3791-b11b-
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Undeleting a Soft-Deleted Entry Using a New RDN

In some cases, the original RDN, `uid=user.1`, may have been allocated to a new user, which is allowed while the entry is in a soft-deleted state. To properly undelete this entry, you need to specify a new RDN value that the entry should be restored with. In this case, specifying the RDN of `uid=user.5` will undelete the original entry but with the new DN in the example below. In addition, the `uid` attribute on the entry will be updated with the new value of `user.5` as well. All other attributes of the users entry including the `entryUUID` will remain unchanged.

To Undelete a Soft-Deleted-Entry Using a New RDN

1. Use `ldapmodify` to undelete a soft-deleted entry that has an original RDN, `uid=user.1`, to a new RDN, `uid=user.5`. You will need to ensure that the DN that you are undeleting the entry to does not already exist as this will lead to an "entry already exists" error if you specify a DN that already exists in the Directory Server as a normal entry.

```
$ bin/ldapmodify --allowUndelete
dn: uid=user.5,ou=People,dc=example,dc=com
changetype:add
ds-undelete-from-dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

2. View the results using `ldapsearch`. You will notice the RDN and the `uid` attribute has changed.

```
dn: uid=user.5,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Aartjan Aalders$59748 Willow Street$Green Bay, TN 66239
postalCode: 66239
description: This is the description for Aartjan Aalders.
uid: user.5
userPassword: {SSHA}RdBCwQ2kIw57LukRthjrFBS/oFylJARnmTnorA==
employeeNumber: 1
initials: AKA
givenName: Aartjan
pager: +1 197 025 3730
mobile: +1 890 430 9077
cn: Aartjan Aalders
sn: Aalders
telephoneNumber: +1 094 100 7524
street: 59748 Willow Street
homePhone: +1 332 432 4295
l: Green Bay
mail: user.3@maildomain.net
st: TN
entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4
```

Modifying a Soft-Deleted Entry

You can modify a soft-deleted entry as you would a regular entry using the `ldapmodify` tool and remains hidden in its soft-deleted state after the change. The only restriction is that you cannot change the DN or run a MODDN operation on the soft-deleted entry.

To move a soft-deleted entry from one machine to another, use the `move-subtree` command and specify the DN of the soft-deleted entry. For more information, see *To Move an Entry from One Machine to Another*.

Note: To modify a soft-deleted entry, the user needs the `soft-delete-read` privilege to access the soft-deleted entry.

To Modify a Soft-Deleted Entry

- Soft-deleted entries can be modified like any regular entry. Use `ldapmodify` and specify the soft-deleted DN.

```
$ bin/ldapmodify
dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
changetype:modify
replace:telephoneNumber
telephoneNumber: +1 390 103 6918
# Processing MODIFY request for entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
# MODIFY operation successful for DN entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Hard Deleting a Soft-Deleted Entry

To permanently remove a soft-deleted entry from the server, you can run `ldapdelete` on the soft-deleted entry for soft-deleted entries. To hard delete a soft-deleted entry, use `ldapdelete` with the --useHardDelete option. The Hard Delete Request Control works with soft deletes. It mostly applies when soft delete policies are in place as a means to override soft deletes requests. If soft deletes are configured, running `ldapdelete` with the Hard Delete Request Control (i.e., using the option, --useHardDelete) guarantees any entry will be permanently deleted.

To Hard Delete a Soft-Deleted Entry (Global Configuration)

- Run `ldapdelete` on a soft-deleted entry to permanently remove it from the Directory Server. This example assumes that you configured soft deletes as a global configuration for requests.

```
$ bin/ldapdelete \
  entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com

Processing DELETE request for entryUUID=4e9b7847-edcb-3791-b11b-
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
DELETE operation successful for DN entryUUID=4e9b7847-edcb-3791-b11b-
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Note:

You cannot soft-delete an already soft-deleted entry. If you use the --useSoftDelete with the `ldapdelete` operation on a soft-deleted entry, an error message will be generated.

```
DELETE operation failed.
Result Code: 53 (Unwilling to Perform)
Diagnostic Message: DELETE operation failed.
```

To Hard Delete a Soft-Deleted Entry (Connection or Request Criteria)

- Run `ldapdelete` with the --useHardDelete option on a soft-deleted entry to permanently remove it from the server. This example assumes that you configured soft deletes using a connection or request criteria.

```
$ bin/ldapdelete --useHardDelete \
  entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Disabling Soft Deletes as a Global Configuration

To disable soft deletes on your Directory Server, simply reset the global configuration property and the remove the soft-delete policy. From that point, all deletes will be processed as hard deletes by default. Any use of the soft-deleted options with the LDAP tools results in an error. Any existing soft-deleted entries that are present after the global configuration is disabled will remain in the server as latent entries.

To Disable Soft Deletes as a Global Configuration

1. Run `dsconfig` to reset the global configuration property. Remember to include the LDAP bind parameters for your system.

```
$ bin/dsconfig set-global-configuration-prop --reset soft-delete-policy
```

2. Run `dsconfig` to delete the soft delete policy that you created. Remember to include the LDAP bind parameters for your system.

```
$ bin/dsconfig delete-soft-delete-policy --policy-name default-soft-delete-
policy
```

Configuring Soft Deletes by Connection Criteria

The Directory Server supports soft deletes where any delete operation is treated as a soft-delete request as long as the LDAP client meets the connection criteria. You can configure soft deletes by defining the connection criteria used in a client connection policy, and then configuring the soft delete connection criteria in the soft-delete policy.

To Enable Soft Deletes by Connection Criteria

1. Configure a soft-delete policy and global configuration as shown in “Configuring Soft Deletes as a Global Configuration”.
2. Create a simple connection criteria using `dsconfig` and name it "Internal Applications". The soft delete connection criteria is configured for a member of a Line of Business (LOB) Applications group connecting from the 10.8.1.0 network.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Internal Applications" \
  --type simple \
  --set included-client-address:10.8.1.0/8 \
  --set "all-included-user-group-dn:cn=LOB
 Applications,ou=Groups,dc=example,dc=com"
```

3. In the soft delete policy created in step 1, set the `auto-soft-delete-connection-criteria` property to the simple criteria created in the previous step.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set "auto-soft-delete-connection-criteria:Internal Applications"
```

To Disable Soft Deletes by Connection Criteria

- To disable soft deletes by connection criteria, simply reset the `auto-soft-delete-connection-criteria` property on the soft-delete policy.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset auto-soft-delete-connection-criteria
```

Configuring Soft Deletes by Request Criteria

Soft deletes can be configured using request criteria within a client connection policy. All delete requests that meet the request criteria is treated as a soft delete. The presence of a soft delete by connection criteria is exclusive of the soft delete by request criteria. Both can be present in a soft-delete policy.

To Enable Soft Deletes by Request Criteria

1. Configure a soft-delete policy and global configuration as shown in “Configuring Soft Deletes as a Global Configuration”.
2. Configure request criteria for soft deletes. The soft delete request criteria is configured for an external delete request from a member of the Internal Applications group matching an entry with object class "inetorgperson" with the request excluding the Soft Delete Request Control and the Hard Delete Request Control.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name "Soft Deletes" \
  --type simple \
  --set "description:Requests for soft delete" \
  --set operation-type:delete \
  --set operation-origin:external-request \
  --set "connection-criteria:Internal Applications" \
  --set not-all-included-request-control:1.3.6.1.4.1.30221.2.5.20 \
  --set "all-included-target-entry-filter:(objectClass=inetorgperson)"
```

3. In the soft delete policy created in step 1, set the `auto-soft-delete-connection-criteria` property to the simple criteria created in the previous step.

```
$ bin/dsconfig create-soft-delete-policy \
  --policy-name default-soft-delete-policy \
  --set "auto-soft-delete-request-criteria:Soft Deletes"
```

To Disable Soft Deletes by Request Criteria

- To disable soft deletes by request criteria, reset the soft-delete policy.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset auto-soft-delete-request-criteria
```

Configuring Soft Delete Automatic Purging

By default, the Directory Server retains soft-deleted entries indefinitely. For companies that want to set up automatic purging of soft-deleted entries, the server provides two properties on the Soft Delete Policy that can be configured for either the maximum retention time for all soft-deleted entries and/or the retained number of soft-deleted entries. These changes take effect without requiring a server restart.

To Configure Soft-Delete Automatic Purging

You can change either the retention time or the retained number of entries to enable automatic purging. By default, both are set to an indefinite retention time and number of entries. The time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), may be preceded by an integer to specify a quantity for that unit, such as "1 d", "52 w", etc. Once you configure the properties, the changes take effect immediately without the need for a server restart.

Note that the server will delete all of the soft-deleted entries according to the policy in effect. If the policy is changed while entries are in the process of being deleted, the new policy takes effect after the in-process batch of entries is deleted and applies to any remaining soft-deleted entries going forward according to the new policy.

1. Retrieve the name of the Soft Delete Policy in effect using the `dsconfig` command. For this example, the Soft Delete Policy is called `default-soft-delete-policy`.

```
$ bin/dsconfig get-global-configuration-prop \
  --property soft-delete-policy
```

2. Do one or both of the following:
   - Run `dsconfig` to set the retention time for soft-deleted entries.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set "soft-delete-retention-time:52 w"
```

- Run `dsconfig` to set the retained number of soft-deleted entries.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set soft-delete-retain-number-of-entries:1000000
```

3. The Soft Delete Policy must be assigned to the global configuration if it has not been assigned yet.

```
$ bin/dsconfig set-global-configuration-prop \
  --set soft-delete-policy:default-soft-delete-policy
```

To Disable Soft-Delete Automatic Purging

You can disable Soft-Delete automatic purging using the `dsconfig` command. The change takes effect immediately without the need of a server restart. However, if the server is in the middle of an automatic soft-delete purging, it may continue to purge entries until the next time it evaluates the Soft Delete Policy.

- Run `dsconfig` to reset the Soft-Delete Policy properties that control automatic purging: `soft-delete-retention-time` and `soft-delete-retain-number-of-entries`.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset soft-delete-retention-time \
  --reset soft-delete-retain-number-of-entries
```

Summary of Soft and Hard Delete Processed

The following table summarizes the resulting actions of a DELETE operation for soft deletes.

Table 7: If No Automatic Soft Delete Criteria is Configured

Action	Result
Delete	Performs a hard delete on the entry.
Delete with the Soft Delete Request Control	Performs a soft delete on the entry.
Delete of a soft-deleted entry	Performs a hard delete on the entry.
Delete of a soft-deleted entry with the Soft Delete Request Control	Not allowed. Generates an `UNWILLING_TO_PERFORM` error.
Delete with a Hard Delete Request Control	Performs a hard delete on the entry.
Delete of a soft-deleted entry with the Hard Delete Request Control.	Performs a hard delete on the entry.

The following table summarizes the resulting actions of a DELETE operation for soft deletes configured by connection criteria or request criteria.

Table 8: If Soft Delete Connection or Request Criteria is Configured

Action	Result
Delete not matching criteria	Performs a hard delete on the entry.
Delete matching criteria	Performs a soft delete on the entry.
Delete of a soft-deleted entry not matching criteria	Performs a hard delete on the entry.
Delete of a soft-deleted entry matching criteria	Performs a hard delete on the entry.
Delete not matching criteria with the Hard Delete Request Control	Performs a hard delete on the entry.
Delete matching criteria with the Hard Delete Request Control	Performs a hard delete on the entry.
Delete with Soft and Hard Delete Request Controls	Not allowed. Generates an `UNWILLING_TO_PERFORM` error.

Summary of Soft Delete Controls and Tool Options

The following table shows the OIDs for each soft delete control. The Soft Delete OIDs are defined in the LDAP SDK generated API documentation.

Table 9: SOft Delete OIDs

OID Type	OID
Soft Delete Request Control	1.3.6.1.4.1.30221.2.5.20
Soft Delete Response Control	1.3.6.1.4.1.30221.2.5.21
Hard Delete Request Control	1.3.6.1.4.1.30221.2.5.22
Soft Undelete Request Control	1.3.6.1.4.1.30221.2.5.23
Soft Delete Entry Access Control	1.3.6.1.4.1.30221.2.5.24

The following table shows the new tool options available for the Soft Delete operations.

Table 10: Soft Delete Tool Options

Action	Result
ldapdelete / ldapmodify	**--useSoftDelete/-s**. Process DELETE operations with the Soft Delete Request Control, whereby entries are renamed, and hidden instead of being permanently deleted. The Directory Server must be configured to allow soft deletes. Note that any entries in the LDIF file with the changetype of `delete` will be processed as a soft-delete request.

Action	Result
ldapdelete	**--useHardDelete**. Process DELETE operations with the Hard Delete Request Control, which bypasses any soft delete policies and processes the delete request immediately without retaining the entry as a soft-deleted entry. The Directory Server must be configured to allow soft deletes.
ldapsearch	**--includeSoftDeletedEntries {with-non-deleted-entries | without-non-deleted-entries | deleted-entries-in-undeleted-form}**. Process search operations with the Soft Delete Entry Access Control. Soft delete search options are as follows: • **with-non-deleted-entries**. Returns all entries matching the search criteria with the results including non-deleted and soft-deleted entries. • **without-non-deleted-entries**. Returns only soft-deleted entries matching the search criteria. • **deleted-entries-in-undeleted-form**. Returns only soft-deleted entries matching the search criteria with the results returned in their undeleted entry form. Users must have access to the Soft Delete Entry Access Control to be able to search for soft-deleted entries.
ldapmodify	**--allowUndelete**. Process ADD operations which include the `ds-undelete-from-dn` attribute as undelete requests. Undelete requests re-add previously soft-deleted entries back to the server as non-deleted entries by providing the Undelete Request Control with the ADD operation. The Directory Server must be configured to allow soft deletes to process any undelete requests and the client user must have the `soft-delete-read` privilege.

The following table shows the symbolic names that can be used with the server's LDAP commands using the --control/-J option.

Table 11: Soft Delete OID Symbolic Names using with the --control/-J Option

Control	Symbolic Name
Soft Delete Request Control	softdelete
Hard Delete Request Control	harddelete
Soft Undelete Request Control	undelete
Soft Delete Entry Access Control	softdeleteentryaccess

Monitoring Soft Deletes

The Directory Server provides monitoring entries and logs to track all soft delete operations. The access and debug logs do not have any options specific for soft deletes.

New Monitor Entries

Two new monitor entries are present for a backend monitor entry. Administrators will see the following additional monitor entries on `cn=userRoot Backend,cn=monitor`:

- **ds-soft-delete-entry-operations-count**. Displays the number of soft-deletes performed on the backend since server startup.
- **ds-undelete-operations-count**. Displays the number of undeletes performed on the backend since server startup.
- **ds-backend-soft-deleted-entry-count**. Displays the current number of soft-deleted entries in the database.
- **ds-auto-purged-soft-deleted-entry-count**. Displays the current number of soft-deleted entries purged since the backend or server was restarted.

To Monitor Soft Deletes

- Run `ldapsearch` on the `cn=userRoot Backend,cn=monitor` branch. Use a search criteria targeting the `ds-backend-monitor-entry` object class.

```
$ bin/ldapsearch --baseDN "cn=userRoot Backend,cn=monitor" \
  --searchScope sub "(objectclass=ds-backend-monitor-entry)"
```

```
dn: cn=userRoot Backend,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-backend-monitor-entry
objectClass: extensibleObject
cn: userRoot Backend
ds-backend-id: userRoot
ds-backend-base-dn: dc=example,dc=com
ds-backend-is-private: FALSE
ds-backend-entry-count: 200001
ds-backend-soft-deleted-entry-count: 1000
ds-soft-delete-operations-count: 40
ds-undelete-operations-count: 20
ds-auto-purged-soft-deleted-entry-count: 0
ds-base-dn-entry-count: 200001 dc=example,dc=com
ds-backend-writability-mode: enabled
```

Access Logs

The access log records the LDAP operations corresponding to soft delete and undelete for DELETE, SEARCH, MODIFY, and ADD operations with the related soft-deleted values. The access log does not require any configuration for soft delete.

For DELETE (soft-delete) operations, the access log displays:

```
[14/May/2012:09:40:16.942 -0500] DELETE RESULT conn=18 op=1 msgID=2
dn="uid=user.1,ou=People,dc=example,dc=com" resultCode=0 etime=30.367
softDeleteEntryDN="entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4+uid=user.1,
ou=People,dc=example,dc=com"
```

For SEARCH operations for soft-deleted entries, the log displays:

```
[14/May/2012:09:40:52.320 -0500] SEARCH RESULT conn=19 op=1 msgID=2
base="dc=example,dc=com" scope=2 filter="(objectclass=ds-soft-delete-entry)"
attrs="ALL" resultCode=0 etime=1.631 entriesReturned=1
```

For MODIFY operations of soft-deleted entries, the log displays:

```
[14/May/2012:09:42:43.679 -0500] MODIFY RESULT conn=20 op=1 msgID=1
dn="entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=exam-
ple,dc=com" resultCode=0 etime=2.639 changeToSoftDeletedEntry=true
```

For ADD (soft undelete) operations, the log displays:

```
[14/May/2012:09:58:16.728 -0500] ADD RESULT conn=25 op=1 msgID=1
dn="uid=user.0,ou=People,dc=example,dc=com" resultCode=0 etime=22.700
```

```
undeleteFromDN="entryUUID=ad55a34a-763f-358f-93f9-da86f9ecd9e4+uid=user.0,
ou=People,dc=example,dc=com"
```

Audit Logs

The audit log captures any MODIFY and DELETE operations of soft-deleted entries. These changes are recorded as fully commented-out audit log entries. The audit log does not require any configuration for soft deletes.

For any soft-deleted entry, the audit log entry displays the `ds-soft-delete-entry-dn` property and its soft-deleted entry DN.

```
# 14/May/2012:10:57:09.054 -0500; conn=30; op=1
# ds-soft-delete-entry-dn: entryUUID=68147342-1f61-3465-8489-
3de58c532130+uid=user.2,ou=People,dc=example,dc=com
dn: uid=user.2,ou=People,dc=example,dc=com
changetype: delete
```

For any MODIFY changes made, the log displays the LDIF, the modifier's name and update time.

```
# 14/May/2012:10:58:33.566 -0500; conn=33; op=1
# dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=People,dc=exam-
ple,dc=com
# changetype: modify
# replace: homePhone
# homePhone: +1 003 428 0966
#-
# replace: modifiersName
# modifiersName: uid=admin,dc=example,dc=com
#-
# replace: modifyTimestamp
# modifyTimestamp: 20131010020345.546Z
```

For any undelete of a soft-deleted entry, the log displays the `ds-undelete-from-dn` attribute plus the entry unique ID, create time and creator's name.

```
# 14/May/2012:10:59:21.754 -0500; conn=34; op=1
dn: uid=user.2,ou=People,dc=example,dc=com
changetype: add
uid: user.2
ds-undelete-from-dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=Peo-
ple,dc=example,dc=com
ds-entry-unique-id:: vw1jg801S7GWrTiS3UE5DA==
createTimestamp:: 20131010181148.630Z
creatorsName: uid=admin,dc=example,dc=com
```

For hard (permanent) deletes of a soft-deleted entry, the log displays the soft-deleted entry DN that was removed.

```
# 14/May/2012:11:00:14.055 -0500; conn=36; op=1
# dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=People,dc=exam-
ple,dc=com
# changetype: delete
```

To Configure the File-Based Audit Log for Soft Deletes

1. Enable the audit log if it is disabled.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
--set enabled:true
```

2. View the audit log. The `soft-delete-entry-audit-behavior` property is set to "`commented`" by default and provides additional information in comments about the soft-deleted entry that was either created or undeleted.

```
# 11/May/2012:15:33:17.552 -0500; conn=13; op=1
# ds-soft-delete-entry-dn:entryUUID=54716bfd-fbc4-3108-ac37-
bf6b1b166e37+uid=user.15,ou=People,dc=example,dc=com
dn: uid=user.15,ou=People,dc=example,dc=com
changetype: delete
```

Change Log

The change log can be configured to capture soft-delete changes to entries, so that external clients, such as a PingDataSync Server, can access these changes. The `ds-soft-delete-entry` attribute represents an entry that has been soft-deleted and is part of the source entry passed into the changelog to indicate the entry has been soft-deleted.

Two important points about soft deletes and the changelog are as follows:

- All soft-delete operations appear in the changelog and appear as a regular DELETE operation. When a soft delete occurs, the resulting changelog entry will include a `ds-soft-delete-entry-dn` operational attribute with the value of the soft-deleted entry DN. If you are using the PingDataSync Server, it does recognize the `ds-soft-delete-entry-dn` attribute and does not do anything with it.
- The changelog backend's `soft-delete-entry-included-operation` property determines whether or not MODIFY or DELETE operations of soft-deleted entries appear in the changelog. By default, the property is not enabled by default.

To Configure Soft Deletes on the Changelog Backend

1. Configure soft deletes on the changelog backend.

```
$ bin/dsconfig set-backend-prop \
--backend-name changelog \
--set soft-delete-entry-included-operation:delete \
--set soft-delete-entry-included-operation:modify
```

2. Run a soft-delete operation on an entry.
3. View the changelog for the soft-deleted entry.

```
$ bin/ldapsearch --baseDN cn=changelog \
  "(objectclass=*)" "+"
```

```
dn: cn=changelog
subschemaSubentry: cn=schema
entryUUID: 9920f7e9-5a04-392a-82a8-32662d7d3863
ds-entry-checksum: 304022441
dn: changeNumber=1,cn=changelog
targetUniqueId: 94f634df-c90e-39aa-bd4a-9183c29746d0
changeTime: 20120511154141Z
ds-soft-delete-entry-dn: entryUUID=94f634df-c90e-39aa-bd4a-
9183c29746d0+uid=user.9,ou=People,dc=example,dc=com
modifyTimestamp: 20131010020345.546Z
createTimestamp:: 20131010181148.630Z
localCSN: 000001373C900852000000000003
modifiersName: uid=admin,dc=example,dc=com
entry-size-bytes: 298
subschemaSubentry: cn=schema
entryUUID: 459b06c6-89f3-307e-a515-22433eb420b6
createTimestamp: 20120511154141.431Z
modifyTimestamp: 20120511154141.431Z
ds-entry-checksum: 1157320579
```

Chapter 8

Importing and Exporting Data

Topics:

The PingDirectory Server supports import or export of the database backends in LDAP Data Interchange Format (LDIF). The `bin/import-ldif` and `bin/export-ldif` tools can be used to create or export database backends for online of offline servers. The tools support options that can restrict the input or output to a subset of the entries or a subset of the attributes within entries. The tools also provide features to compress, encrypt or digitally sign the data.

This chapter presents the following topics:

Importing Data

The PingDirectory Server provides initialization mechanisms to import database files. The `import-ldif` command-line tool imports data from an LDAP Data Interchange Format (LDIF) file. The data imported by the `import-ldif` command can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the LDIF file. The command also supports importing data that has been compressed, encrypted or digitally signed or both.

The `import-ldif` utility can be run with the server offline or online. If the server is online, administrators can initiate the import from a local or remote client. The LDIF file that contains the import data must exist on the server system. During an online import, the target database repository, or backend, will be removed from service and data held in that backend will not be available to clients.

The `import-ldif` tool has been modified to help guard against accidental overwriting of existing backend data with the addition of the --overwriteExistingEntries option. This option must be present when performing an import into a backend with a branch that already contains entries (although the option is not needed if a branch contains just a single base entry).

Validating an LDIF File

Prior to importing data, you can validate an import file using the Directory Server's `validate-ldif` tool. When run, the tool binds to the Directory Server, locally or remotely, and validates the LDIF file to determine whether it violates the server's schema. Those elements that do not conform to the schema will be rejected and written to standard output. You can specify the path to the output file to which the rejected entries are written and the reasons for their rejection. The `validate-ldif` tool works with regular non-compressed LDIF files or gzip-compressed LDIF files.

To process large files faster, you can also set the number of threads for validation. The tool also provides options to skip specified schema elements if you are only validating certain items, such as attributes only. Use the --help option to view the arguments.

To Validate an LDIF File

- Use the `validate-ldif` tool to validate an LDIF file. Make sure the server is online before running this command.

```
$ bin/validate-ldif --ldifFile /path/to/data.ldif \
  --rejectFile rejectedEntries
```

```
1 of 200 entries (0 percent) were found to be invalid.
1 undefined attributes were encountered.
Undefined attribute departmentname was encountered 1 times.
```

Computing Database Cache Estimate

After successful completion of an import, the `import-ldif` command lists detailed information about the database cache characteristics of the imported data set. The current server configuration is considered along with the capabilities of the underlying hardware to guide decisions for changing JVM size and database-cache-percent for the backend.

The `import-ldif` command will complete with a summary of database cache usage characteristics for the imported data set. Additional files are available in the `/logs/tools` directory that describe the database cache characteristics in more detail.

Tracking Skipped and Rejected Entries

During import, entries can be skipped if they do not belong in the specified backend, or if they are part of an excluded base DN or filter. The --skipFile {path} argument can be used on the command line to indicate that any entries that are skipped should be written to a specified file. You can add a comment indicating why the entries were skipped.

Similarly, the --rejectFile {path} argument can be added to obtain information about which entries were rejected and why. An entry can be rejected if it violates the server's schema constraints, if its parent entry does not exist, if another entry already exists with the same DN, or if it was rejected by a plug-in.

Running an Offline Import

You can run the `import-ldif` tool offline to import LDIF data encoded with the UTF-8 character set. This data can come from LDIF files, compressed LDIF files (GZIP format), or from data generated using a MakeLDIF template. You do not need to authenticate as an administrator when performing offline LDIF imports.

To Perform an Offline Import

- Use the `import-ldif` command to import data from an LDIF file. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --rejectFile /path/to/reject.ldif --skipFile /path/to/skip.ldif
```

To Perform an Offline LDIF Import Using a Compressed File

- Use the `import-ldif` command to import data from a compressed gzip formatted file. You must also use the --isCompressed option to indicate that the input file is compressed. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --isCompressed \
  --ldifFile /path/to/data.gz --rejectFile /path/to/reject.ldif \
  --skipFile /path/to/skip.ldif
```

To Perform an Offline LDIF Import Using a MakeLDIF Template

- Use the `import-ldif` command to import data from a MakeLDIF template, which is located in the `<server-root>/config/MakeLDIF`. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

  The following command uses the standard data template and generates 10,000 sample entries, and then imports the file to the server.

```
$ bin/import-ldif --backendID userRoot \
  --templateFile config/MakeLDIF/example.template
```

Running an Online LDIF Import

Administrators can run LDIF imports while the server is online from another remote server. The online import resembles the offline import, except that you must provide information about how to connect and authenticate to the target server. You can schedule the import of an LDIF file to occur at a particular time using the --task and --start YYYYMMDDhhmmss options of the `import-ldif` tool.

You can also specify email addresses for users that should be notified whenever the import process completes (regardless of success or failure, or only if the import fails). To set up SMTP notifications, see *Working with the SMTP Account Status Notification Handler*.

To Perform an Online LDIF Import

- Use the `import-ldif` tool to import data from an LDIF. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task --hostname server1 --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --backendID userRoot --ldifFile userRoot.ldif
```

To Schedule an Online Import

1. Use the `import-ldif` tool to import data from an LDIF file at a scheduled time. To specify a time in the UTC time zone, include a trailing “Z”. Otherwise, the time will be treated as a local time in the time zone configured on the server. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task \
  --hostname server1 \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password \
  --backendID userRoot \
  --ldifFile /path/to/data.ldif \
  --start 20111026010000 \
  --completionNotify import-complete@example.com \
  --errorNotify import-failed@example.com
```

```
Import task 2011102617321510 scheduled to start Oct 26, 2011 1:00:00 AM CDT
```

2. Confirm that you successfully scheduled your import task using the `manage-tasks` tool to view a summary of all tasks on the system.

```
$ bin/manage-tasks --summary
```

```
ID            Type   Status
-----------------------------------------------
2011102617321510 Import Waiting on start time
```

3. Use the `manage-tasks` tool to monitor the progress of this task. Use the task ID of the import task. If you cannot find the task ID, use the --summary option to view a list of all tasks scheduled on the Directory Server.

```
$ bin/manage-tasks --info 2011102617321510
```

```
Task                    Details
-----------------------------------------------------
ID                      2011102617321510
Type                    Import
Status                  Waiting on start time
Scheduled Start Time    Oct 26, 2011 1:00:00 AM CDT
Actual Start Time
Completion Time
Dependencies Failed     None
Dependency Action       None
Email Upon Completion   admin@example.com
Email Upon Error        admin@example.com

Import        Options
----------------------------
LDIF File  /path/to/data.ldif
Backend ID userRoot
```

To Cancel a Scheduled Import

- Use the `manage-tasks` tool to cancel the scheduled task.

```
$ bin/manage-tasks --cancel 2011102417321510
```

Adding Entries to an Existing Directory Server

To add entries to an existing Directory Server while preserving operational attributes, such as `createTimestamp` or `modifiersName`, the Ignore No User Modification control must be attached to the request. The Ignore No User Modification control allows modification of certain attributes that have the No User Modification constraint. Special care should be used with this control.

The Ignore No User Modification is only applied to ADD requests. Using the control to modify an existing entry, resulting in an operational attribute change, will fail.

To Append Entries to an Existing Directory Server

- Use `ldapmodify` with the Ignore No User Modification control (i.e., the OID is 1.3.6.1.4.1.30221.2.5.5).

```
$ bin/ldapmodify --control 1.3.6.1.4.1.30221.2.5.5 \
  --filename change-record.ldif
```

Filtering Data Import

The `import-ldif` command provides a way to either include or exclude specific attributes or entries during an import. The arguments are summarized as follows:

Table 12: Inclusion and Exclusion Arguments for import-ldif

Argument	Description
--includeBranch	Base DN of a branch to include in the LDIF import (can be specified multiple times)
--excludeBranch	Base DN of a branch to exclude from the LDIF import (can be specified multiple times)
--includeAttribute	Attribute to include in the LDIF import (can be specified multiple times)
--excludeAttribute	Attribute to exclude from the LDIF import (can be specified multiple times)
--includeFilter	Search filter to identify entries to include in the LDIF import (can be specified multiple times)
--excludeFilter	Search filter to identify entries to exclude from the LDIF import (can be specified multiple times)
--excludeOperational	Exclude operational attributes from the LDIF import.
--excludeReplication	Exclude replication attributes from the LDIF import.
--excludeSoftDelete	Exclude soft delete entries from the LDIF import.

Exporting Data

The PingDirectory Server `export-ldif` command-line tool exports data from Directory Server backend to an LDAP Data Interchange Format (LDIF) file. The tool must be run in the non-task based mode, which implies that it works outside of the server JVM process. The `export-ldif` must be run without connection or task arguments while the server is either online or offline. This tool exports a point-in-time snapshot of the backend which is guaranteed to provide a consistent state of the database, in LDIF, which can be reimported with `import-ldif` if necessary.

The data exported by `export-ldif` can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the backend. This is accomplished by specifying branches, filters, and attributes to include or exclude. The exported LDIF can be compressed, encrypted or digitally signed.

Note: LDIF exports can be configured as recurring tasks with `dsconfig create-recurring-task`, and then scheduled to run when added to a recurring task chain.

To Perform an Export

- Use the `export-ldif` command to export data to an LDIF file.

```
$ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif
```

To Perform an Export from Specific Branches

- Use the `export-ldif` command to export data to an LDIF file under a specific branch from the userRoot backend of the local Directory Server into a compressed file. The command also excludes operational attributes from the exported data and wraps long lines at column 80.

```
$ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif.gz --
compress \
  --includeBranch ou=people,dc=example,dc=com --excludeOperational \
  --wrapColumn 80
```

Encrypting LDIF Exports and Signing LDIF Files

The Directory Server provides features to encrypt data during an LDIF export using the `export-ldif` --encryptLDIF option and to allow the encrypted LDIF file to be imported onto the same instance or another server in the same replication topology using the `import-ldif` tool. A `--doNotEncrypt` argument can be used to force an LDIF export to be unencrypted, even if automatic encryption is enabled. The `--maxMegabytesPerSecond` argument can be used to impose a limit on the rate at which the LDIF file may be written to disk.

The `export-ldif` tool can be used with the `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, and `--encryptionSettingsDefinitionID` arguments to specify which key to use for encrypting the export. The `import-ldif tool` will automatically detect encryption and compression, and have `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile` options as well.

The Directory Server also provides an additional argument that digitally signs the contents of the LDIF file, which ensures that the content has not been altered since the export. To digitally sign the contents of the exported LDIF file, use the `export-ldif` --sign option. To allow a signed LDIF file to be imported onto the same instance or another server in the same topology, use the `import-ldif` --isSigned option.

Note that there is not much added benefit to both signing and encrypting the same data, since encrypted data cannot be altered without destroying the ability to decrypt it.

To Encrypt an LDIF Export

- Run `export-ldif` tool with the `--encryptLDIF` option to encrypt the data during an export to an output LDIF file. The following command runs an offline export of the userRoot backend, and encrypts the file when written to an output file called `data.ldif`.

```
$ bin/export-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --encryptLDIF
```

To Import an Encrypted LDIF File

An encrypted LDIF file can be imported into the same instance from which it was exported, or into any other server in the same replication topology with that instance. You cannot import an encrypted LDIF file into a server that is not in some way connected to the instance from which it was exported.

- Run the import-ldif tool to import the encrypted LDIF file from the previous example. The command imports the `data.ldif` file, decrypts the contents while overwriting the existing contents to the `userRoot` backend. The tool automatically determines encryption and compression, and it can automatically identify the correct key for exports that were encrypted with a key obtained from an encryption settings definition or an internal topology key.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries
```

To Sign an Export

- Run `export-ldif` tool with the `--sign` option to digitally sign the data during an export to an output LDIF file. The following command runs an offline export of the `userRoot` backend, and signs the content when written to an output file called `data.ldif`.

```
$ bin/export-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif --sign
```

To Import a Signed LDIF File

- Run the `import-ldif` tool to import the signed LDIF file from the previous example. The command imports the `data.ldif` file, checks the signature of the contents while overwriting the existing contents to the `userRoot` backend. The command requires the --isSigned option, which instructs the tool that the contents of the LDIF file is signed.

```
$ bin/import-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries --isSigned
```

Filtering Data Exports

The `export-ldif` command analogous arguments to the `import-ldif` tool to provide a way to either include or exclude specific attributes or entries during an export. The arguments are summarized as follows:

Table 13: Inclusion and Exclusion Arguments for export-ldif

Argument	Description
--includeBranch	Base DN of a branch to include in the LDIF export (can be specified multiple times)
--excludeBranch	Base DN of a branch to exclude from the LDIF export (can be specified multiple times)
--includeAttribute	Attribute to include in the LDIF export (can be specified multiple times)
--excludeAttribute	Attribute to exclude from the LDIF export (can be specified multiple times)
--includeFilter	Filter to identify entries to include in the LDIF export (can be specified multiple times)
--excludeFilter	Filter to identify entries to exclude from the LDIF export (can be specified multiple times)
--excludeOperational	Exclude operational attributes from the LDIF export.
--excludeReplication	Exclude replication attributes from the LDIF export.
--excludeSoftDelete	Exclude soft delete entries from the LDIF export.

Scrambling Data Files

The Directory Server `transform-ldif` tool provides backward compatibility with the former `scramble-ldif` tool, with additional functionality for configuring input and output files. The `transform-ldif` tool reads data from one or more source LDIF files and writes the transformed data to a single output file.

Using this tool to scramble data, enables obscuring the values of certain attributes so that it is difficult to determine the original values in the source data, while also preserving the characteristics of the associated attribute syntax. This process is repeatable, so that if the same value appears multiple times, it will yield the same scrambled representation each time. Scrambling can be applied to both LDIF entries and LDIF change records.

The process of scrambling data is not the same as encryption. It should only be used to provide simple obfuscation of data. The following are general guidelines for scrambling attributes:

- If the attribute is `userPassword` and its value starts with a scheme name surrounded by curly braces, such as "{SSHA256}XrgyNdl3fid7KYdhd/Ju47KJQ5PYZqlUlyzxQ28f/QXUnNd9fupj9g==", the scheme name will be left unchanged and the rest of the value will be treated like a generic string.
- If the attribute is `authPassword` and its value contains at least two dollar signs, such as "SHA256$QGbHtDCi1i4=$8/X7XRGaFCovC5mn7ATPDYlkVoocDD06Zy3lbD4AoO4=", the portion up to the first dollar sign (which represents the name of the encoding scheme) is preserved and the remainder of the value is treated like a generic string.
- If an attribute has a Boolean syntax, the scrambled value will be either `TRUE` or `FALSE`. The determination to use a value of `TRUE` or `FALSE` is random, so scrambling Boolean values is not repeatable. By randomizing the scrambling for Boolean values, the syntax and obfuscation of the original value is preserved.
- If an attribute has a distinguished name syntax (or a related syntax, such as a name and optional UID), scrambling is applied to the values of RDN components for any attributes to be scrambled. For example, if the tool is configured to scramble both the `member` and `uid` attributes, and an entry has a member attribute with a value of "uid=john.doe,ou=People,dc=example,dc=com", that member value will be scrambled in a way that only obscures the "john.doe" portion but leaves the attribute names and all values of non-scrambled attributes intact.
- If an attribute has a generalized time syntax, that value is replaced with a randomized timestamp using the same format (the same number of digits and the same time zone indicator). The randomization will be over a time range that is double the difference between the time the `transform-ldif` tool was launched and the timestamp to be scrambled. For values where that time difference is less than one day, one day will be added to the difference before it is doubled.
- If an attribute has an integer, numeric string, or telephone number syntax, scrambling is only applied to numeric digits while all other characters are left intact. If there are multiple digits, then the first digit will be nonzero.
- If an attribute has an octet string syntax, it is scrambled as follows:
  - Each byte that represents a lowercase ASCII letter is replaced with a randomly-selected lowercase ASCII letter.
  - Each byte that represents a uppercase ASCII letter is replaced with a randomly-selected uppercase ASCII letter.
  - Each byte that represents an ASCII digit is replaced with a randomly-selected ASCII digit.
  - Each byte that represents a printable ASCII symbol is replaced with a randomly-selected printable ASCII symbol.
  - Each byte that represents an ASCII control character is replaced with a randomly-selected ASCII letter, digit, or symbol.
  - Each non-ASCII byte will be replaced with a randomly-selected non-ASCII byte.
- If an attribute has a value that represents a valid JSON object, the resulting value will also be a JSON object. All field names will be left intact, and only the values of those fields may be scrambled. If the `--scrambleJSONField` argument is provided, only the specified fields will have values scrambled. Otherwise, the values of all fields will be scrambled. Field values are scrambled as follows:
  - Null values are not scrambled.
  - Boolean values are replaced with randomly-selected Boolean values. As with attributes with a Boolean syntax, these values are non-repeatable.
  - Number values will have only their digits replaced with randomly-selected digits and all other characters (minus sign, decimal point, exponentiation indicator) are left unchanged.
  - String values will be replaced with a randomly-selected generic string.
  - Array values have scrambling applied as appropriate for each value in the array. If the array field itself should be scrambled, then all values in the array are scrambled. Otherwise, only JSON objects contained inside the array have scrambling applied to appropriate fields.

  - JSON values have scrambling applied as appropriate for their fields.
- If an attribute does not match any of the previous criteria, it is scrambled as follows:
  - Each lowercase ASCII letter is replaced with a randomly-selected lowercase ASCII letter.
  - Each uppercase ASCII letter replaced with a randomly-selected uppercase ASCII letter.
  - Each ASCII digit is replaced with a randomly-selected ASCII digit.
  - All other characters are left unchanged.

The following example reads from an LDIF file named `original.ldif`, scrambles the values of the `telephoneNumber`, `mobile`, and `homeTelephoneNumber` attributes, and writes the results to `scrambled.ldif`:

```
$ bin/transform-ldif --sourceLDIF original.ldif \
  --targetLDIF scrambled.ldif \
  --scrambleAttribute telephoneNumber \
  --scrambleAttribute mobile \
  --scrambleAttribute homeTelephoneNumber \
  --randomSeed 0
```

Chapter

9

Backing Up and Restoring Data

Topics:

The PingDirectory Server provides efficient `backup` and `restore` command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based cron scheduler or using the Directory Server's Task-based scheduler.

This chapter presents the following topics:

Backing Up and Restoring Data

Administrators should have a comprehensive backup strategy and schedule that comprise of daily, weekly, and monthly backups including incremental and full backups of the directory server data, configuration, and backends. Administrators should also have a backup plan for the underlying filesystem. This dual purpose approach provides excellent coverage in the event that a server database must be restored for any reason.

Note: Backups can be configured as recurring tasks with `dsconfig create-recurring-task`, and then scheduled to run when added to a recurring task chain.

The PingDirectory Server provides efficient `backup` and `restore` command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based cron scheduler or using the Directory Server's Task-based scheduler. The Directory Server can run backups with the server online while processing other requests, so that there is no need to shut down the server or place it in read-only mode prior to starting a backup.

If you back up more than one backend, the `backup` tool creates a subdirectory below a specified backup directory for each backend. If you back up only a single backend, then the backup files will be placed in the specified directory. A single directory can only contain files from one backend, so that you cannot have backup files from multiple different backends in the same backup directory.

When performing a backup, the server records information about the current state of the server and backend, including the server product name, the server version, the backend ID, the set of base DNs for the backend, and the Java class used to implement the backend logic. For JE backends, the backup descriptor also includes information about the Berkeley DB JE version and information about the attribute and VLV indexes that have been defined.

When restoring a backup, the server compares the descriptor obtained from the backup with the current state of the server and backend. If any problems are identified, the server generates warnings or errors. The administrator can choose to ignore the warnings with the `ignoreCompatibilityWarnings` option to the `restore` tool, whereas errors will always cause the restore to fail. For example, when restoring a *newer* backup into an older version of the server, a warning will be generated. When restoring an *older* backup into a new version of the server, no warning will be generated, but because the `config` and `schema` backends require special handling, the server generates an error if the server versions do not match exactly (major, minor, point, and patch version numbers).

Both the `backup` and `restore` tools provide encryption options `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, and `--encryptionSettingsDefinitionID` that can be used to specify which key to use for encrypting the backup. For backups encrypted with an encryption settings definition or an internal topology key, the server will automatically determine the correct key. Or, the `--doNotEncrypt` argument can be used to force a backup to be unencrypted even if automatic encryption is enabled.

If needed, the `--maxMegabytesPerSecond` argument can be used to impose a limit on the rate at which the backup may be written to disk.

Retaining Backups

The backup tool can be used with either the `--retainPreviousFullBackupCount` or `--retainPreviousFullBackupAge` arguments to identify which previous backups should be preserved. Any other backups in that directory will be removed. A new backup will always be preserved. If the new backup is an incremental backup, then any other backups it depends on will also be preserved. However, older backups in the same directory are eligible to be removed.

If the `--retainPreviousFullBackupCount` argument is provided, that number of the most recent previous full backups will be preserved (along with any incremental backups that depend on them). Any other previous full backups (and their dependent incremental backups) can be removed. If the `--retainPreviousFullBackupAge` argument is provided, its value must be a duration represented as an integer followed by a time unit. Any full backups (and their dependent incremental backups) created longer ago than that duration will be eligible to be removed. If both the `--retainPreviousFullBackupCount` and `--retainPreviousFullBackupAge` arguments are provided, then only backups that don't satisfy either condition will be deleted. A value of zero can be specified for the `--retainPreviousFullBackupCount` argument so that only the most recent backup is preserved (along with its dependencies), and all previous backups will be removed.

Note: The `remove-backup` tool also supports the `--retainFullBackupCount` and `--retainFullBackupAge` arguments to delete any backups outside the provided retention criteria.

To List the Available Backups on the System

- Use the `restore` tool to list the backups in a `backup` directory.

```
$ bin/restore --listBackups --backupDirectory /mybackups
```

```
[13:26:21] The console logging output is also available in '/
ds/PingDirectory/logs/ tools/restore.log'

Backup ID:          20120212191715Z
Backup Date:        12/Feb/2012:13:17:19 -0600
Is Incremental:     false
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash: false
Has Signed Hash:    false
Dependent Upon:     none
Backup ID:          20120212192411Z
Backup Date:        12/Feb/2012:13:24:16 -0600
Is Incremental:     true
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash: false
Has Signed Hash:    false
Dependent Upon:     20120212191715Z
```

To Back Up All Backends

- Use `backup` to save the all of the server's backends. The optional --compress option can reduce the amount of space that the backup consumes, but can also significantly increase the time required to perform the backup.

```
$ bin/backup --backUpAll --compress --backupDirectory /path/to/backup
```

To Back Up a Single Backend

- Go to the server root directory, and use the `backup` tool to save the single backend, userRoot.

```
$ bin/backup --backendID userRoot --compress --backupDirectory /path/to/
backup
```

To Perform an Offline Restore

- Use the `restore` command to restore the userRoot backend. Only a single backend can be restored at a time. The Directory Server must be shut down before performing an offline restore.

```
$ bin/restore --backupDirectory /path/to/backup/userRoot
```

Note: The server root directory should never be restored from a file system backup or snapshot.

To Assign an ID to a Backup

- Go to the server root directory, and use the backup tool to save the single backend, userRoot. The following command assigns the backup ID "weekly" to the userRoot backup. The backup file appears under `backups/userRoot` directory as `userRoot-backup-weekly`.

```
$ bin/backup --backupDirectory /path/to/backups/userRoot \
  --backendID userRoot --backupID weekly
```

To Run an Incremental Backup on All Backends

The Directory Server provides support for incremental backups, which backs up only those items that have changed since the last backup (whether full or incremental) on the system, or since a specified earlier backup. Incremental backups must be placed in the same backup directory as the full backup on which they are based.

Not all backends support incremental backups. If a backend does not support incremental backups, use of the `--incremental` option will have no effect, and a full backup will be taken.

- The following command runs an incremental backup on all backends based on the most recent backup:

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup
```

To Run an Incremental Backup on a Single Backend

- Go to the server root directory, and use `backup` to save the single backend, userRoot.

```
$ bin/backup --backendID userRoot --incremental --backupDirectory /path/to/
backup
```

To Run an Incremental Backup based on a Specific Prior Backup

- You can run an incremental backup based on a specific prior backup that is not the most current version on the system. To get the backup ID, use the `restore` --listBackups command (see below).

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup \
  --incrementalBaseID backup-ID
```

To Restore an Incremental Backup

The process for restoring an incremental backup is exactly the same as the process for restoring a full backup for both the online and offline restore types. The `restore tool` will automatically ensure that the full backup and any intermediate incremental backups are restored first before restoring the final incremental backup. The tool will not restore any files in older backups that are no longer present in the final data set.

To Schedule an Online Backup

You can schedule a backup to run as a Task by specifying the timestamp with the --task and --start options. The option is expressed in "YYYYMMDDhhmmss'" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

For online (remote) backups, the backup operation can be conducted while the PingDirectory Server is online if you provide information about how to connect and to authenticate to the target Directory Server.

- You can schedule the backup to occur at a specific time using the Task-based --start YYYYMMDDhhmmss option. To specify a time in the UTC time zone format, add a trailing “Z” to the time. Otherwise, the time will be treated as a local time in the time zone configured on the server.

```
$ bin/backup --backUpAll --task --start 20111025010000 \
  --backupDirectory /path/to/backup --completionNotify admin@example.com \
  --errorNotify admin@example.com
```

```
Backup task 2011102500084110 scheduled to start Oct 28, 2011 1:00:00 AM CDT
```

To Schedule an Online Restore

By providing connection and authentication information (and an optional start time), the restore can be performed via the Tasks subsystem while the server is online. The Tasks subsystem allows you to schedule certain operations, such as `import-ldif`, `backup`, `restore`, `start-server`, and `stop-server`. You can schedule a restore to run as a Task by specifying the timestamp with the --task and --start options. The option is expressed in "YYYYMMDDhhmmss'" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

- The backend that is being restored will be unavailable while the restore is in progress. To specify a time in the UTC time zone, add a trailing "Z" to the time. Otherwise, the time will be treated as a local time in the configured time zone on the server.

```
$ bin/restore --task --start 20111025010000 \
  --backupDirectory /path/to/backup/userRoot \
  --completionNotify admin@example.com --errorNotify admin@example.com
```

To Encrypt a Backup

- Go to the server root directory, and use the `backup` tool to backup up the single backend, userRoot and encrypt it with the `--encrypt` option.

```
$ bin/backup --encrypt --backendID userRoot --compress --backupDirectory /
path/to/backup
```

To Sign a Hash of the Backup

- Go to the server root directory, and use the `backup` tool to backup up the single backend, userRoot. Use the `–signHash` option to generate a hash of the backup contents and digitally sign the hash of the backup contents. If you want to generate only a hash of the backup contents, run backup with the `--hash` option.

```
$ bin/backup --signHash --backupDirectory backups/userRoot --backendID
 userRoot \
  --backupDirectory /path/to/backup
```

To Restore a Backup

- Go to the server root directory, and use the `restore` tool to restore a backup. The `backup` tool uses a descriptor file to access property information used for the backup, indicating if the backup was compressed, signed and/or encrypted.

```
$ bin/restore --backupDirectory /path/to/backup
```

Moving or Restoring a User Database

Part of any disaster recovery involves the restoration of the user database from one server to another. You should have a well-defined backup plan that takes into account whether or not your data is replicated among a set of servers. The plan is the best insurance against significant downtime or data loss in the event of unrecoverable database issue.

Keep in mind the following general points about database recovery:

- **Regular Backup from Local Replicated Directory Server**. Take a backup from a local replicated directory server and restore to the failed server. This will be more recent than any other backup you have.
- **Restore the Most Recent Backup**. Restore the most recent backup from a local server. In some cases, this may be preferred over taking a new backup if that would adversely impact performance of the server being backed up although it will take longer for replication to play back changes.

- **Contact Support**. If all else fails, contact your authorized support provider and they can work with you (and possibly in cooperation with the Oracle Berkeley DB JE engineers) to try a low-level recovery, including tools that attempt to salvage whatever data they can obtain from the database.

Comparing the Data in Two Directory Servers

The PingDirectory Server provides an `ldap-diff` tool to compare the data on two LDAP servers to determine any differences that they may contain. The differences are identified by first issuing a subtree search on both servers under the base DN using the default search filter `(objectclass=*)` to retrieve the DNs of all entries in each server. When the tool finds an entry that is on both servers, it retrieves the entry from each server and compares all of its attributes. The tool writes any differences it finds to an LDIF file in a format that could be used to modify the content of the source server, so that it matches the content of the target server. Any non-synchronized entries can be compared again for a configurable number of times with an optional pause between each attempt to account for replication delays.

You can control the specific entries to be compared with the --searchFilter option. In addition, only a subset of attributes can be compared by listing those attributes as trailing arguments of the command. You can also exclude specific attributes by prepending a ^ character to the attribute. (On Windows operating systems, excluded attributes must be quoted, for example, "`^attrToExclude`".) The `@objectClassName` notation can be used to compare only attributes that are defined for a given objectclass.

The `ldap-diff` tool can be used on servers actively being modified by checking differing entries multiple times without reporting false positives due to replication delays. By default, it will re-check each entry twice, pausing two seconds between checks. These settings can be configured with the --numPasses and --secondsBetweenPass options. If the utility cannot make a clean comparison on an entry, it will list any exceptions in comments in the output file.

The Directory Server user specified for performing the searches must be privileged enough to see all of the entries being compared and to issue a long-running, unindexed search. For the Directory Server, the out-of-the-box `cn=Directory Manager` user has these privileges, but you can assign the necessary privileges by setting the following attributes in the user entry:

```
ds-cfg-default-root-privilege-name: unindexed-search
ds-cfg-default-root-privilege-name: bypass-acl
ds-rlim-size-limit: 0
ds-rlim-time-limit: 0
ds-rlim-idle-time-limit: 0
ds-rlim-lookthrough-limit: 0
```

The `ldap-diff` tool tries to make efficient use of memory, but it must store the DNs of all entries in memory. For Directory Servers that contain hundreds of millions of entries, the tool might require a few gigabytes of memory. If the progress of the tool slows dramatically, it might be running low on memory. The memory used by the `ldap-diff` tool can be customized by editing the `ldap-diff.java-args` setting in the `config/java.properties` file and running the `dsjavaproperties` command.

If you do not want to use a subtree search filter, you can use an input file of DNs for the source, target, or both. The format of the file can accept various syntaxes for each DN:

```
dn: cn=this is the first dn
dn: cn=this is the second dn and it is wrapped cn=this is the third dn
# The following DN is base-64 encoded dn::
Y249ZG9uJ3QgeW91IGhhdmUgYmV0dGVyIHRoaW5ncyB0byBkbyB0aGFuIHNlZSB3aGF0IHRoaXMgc2F5cw==
# There was a blank line above dn: cn=this is the final entry.
```

Caution: Do not manually update the servers when the tool identifies differences between two servers involved in replication. First contact your authorized support provider for explicit confirmation, because manual updates to the servers risk introducing additional replication conflicts.

To Compare Two Directory Servers Using ldap-diff

1. Use `ldap-diff` to compare the entries in two Directory Server instances. Ignore the `userpassword` attribute due to the one-way password hash used for the password storage scheme.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN dc=example,dc=com --searchFilter "(objectclass=*)"
 "^userpassword"
```

2. Open the output file in a text editor to view any differences. The file is set up so that you can re-apply the changes without any modification to the file contents. The file shows any deletes, modifies, and then adds from the perspective of the source server as the authoritative source.

```
# This file contains the differences between two LDAP servers.
#
# The format of this file is the LDIF changes needed to bring server
# ldap://server1.example.com:1389 in sync with server
# ldap://server2.example.com:2389.
#
# These differences were computed by first issuing an LDAP search at both
# servers under base DN dc=example,dc=com using search filter
 (objectclass=*)
# and search scope SUB to first retrieve the DNs of all entries. And then
 each
# entry was retrieved from each server and attributes: [^userpassword] were
# compared. # # Any entries that were out-of-sync were compared a total of 3
 times
# waiting a minimum of 2 seconds between each attempt to account for
 replication
# delays.
#
# Comparison started at [24/Feb/2010:10:34:20 -0600]
# The following entries were present only on ldap://server2.example.com:2389
 and
# need to be deleted. This entry existed only on ldap://
server1.example.com:1389
# Note: this entry might be incomplete. It only includes attributes:
# [^userpassword]dn: uid=user.200,ou=People,dc=example,dc=com
# objectClass: person
# objectClass: inetOrgPerson
... (more attributes not shown) ...
# st: DC
dn: uid=user.200,ou=people,dc=example,dc=com
changetype: delete

# The following entries were present on both servers but were out of sync.

dn: uid=user.199,ou=people,dc=example,dc=com
changetype: modify
add: mobile
mobile: +1 300 848 9999
-
delete: mobile
mobile: +1 009 471 1808

# The following entries were missing on ldap://server2.example.com:2389 and
 need
# to be added. This entry existed only on ldap://server2.example.com:2389
# Note: this entry might be incomplete. It only includes attributes:
```

```
# [^userpassword]
dn: uid=user.13,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
... (more attributes not shown) ...
# Comparison completed at [24/Feb/2010:10:34:25 -0600]
```

To Compare Configuration Entries Using ldap-diff

- Use `ldap-diff` to compare the configuration entries in two Directory Server instances. The filter searches all configuration entries. Ignore the `userpassword` attribute due to the password storage scheme that uses a one-way hashing algorithm.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN cn=config --searchFilter "(objectclass=*)" "^userpassword"
```

To Compare Entries Using Source and Target DN Files

- Use `ldap-diff` to compare the entries in two Directory Server instances. In the following example, the utility uses a single DN input file for the source and target servers, so that no search filter is used. Ignore the `userpassword` attribute due to the password storage scheme that uses a one-way hashing algorithm.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN "dc=example,dc=com" --sourceDNsFile input-file.ldif \
  --targetDNsFile input-file.ldif "^userpassword"
```

To Compare Directory Servers for Missing Entries Only Using ldap-diff

- Use `ldap-diff` to compare two Directory Servers and return only those entries that are missing on one of the servers using the --missingOnly option, which can significantly reduce the runtime for this utility.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN dc=example,dc=com --searchFilter "(objectclass=*)"
 "^userpassword" \
  --missingOnly
```

Chapter 10

Working with Indexes

Topics:

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server.

This chapter presents topics related to indexes:

Overview of Indexes

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server. The Directory Server automatically creates index files when you first initialize a base DN or when you use the `dsconfig` tool to create a local DB backend. During modify operations, the Directory Server updates the database index files. If encryption is enabled on the server, indexes will be encrypted.

The PingDirectory Server comes with the following types of indexes:

- **Default system indexes** to ensure that the server operates efficiently. Indexes consist of database files that contain index keys mapping to the list of entry IDs.
- **Default Local DB indexes** that are created for each database suffix. Modify the index to meet your system's requirements using the `dsconfig` tool.
- **Local DB VLV indexes** that allow a client to request the server to send search results using the Virtual List View control.
- **Filtered Indexes** that provide the ability to index an attribute but only for entries that match a specified filter based on an equality index. The filtered index can only be used for searches containing that filter. The filtered index can be maintained independently of the equality index for that attribute and even if a normal equality index is not maintained for that attribute.

General Tips on Indexes

Administrators should keep the following tips in mind when working with indexes:

- **Important Critical Indexes**. The Directory Server has several built-in indexes on the Local DB Backend that are critical to internal server processing and should never be removed.

  ```
  aci, ds-entry-unique-id, objectClass
  ```
- **Built-in Indexes for Efficient Queries**. The Directory Server has built-in indexes on the Local DB Backend. Internal processing of the server relies on the `aci`, `ds-soft-delete-from-dn`, `ds-soft-delete-timestamp`, `entryUUID`, `member`, `objectClass`, and `uniqueMember` indexes, which must not be removed. The `mail` and `uid` indexes can be removed, but these attributes are referenced from the Password Modify Extended Operation and will cause problems with components such as the Exact Match Identity Mapper. If the `mail` or `uid` indexes are removed, additional configuration changes may be necessary to ensure that the server starts properly. The `cn`, `givenName`, `mail`, `sn`, and `telephoneNumber` indexes can be safely removed if clients do not query on these attributes. This will reduce the size of the database both on disk and in memory.
- **Online Rebuilds**. Whenever an online index rebuild is in progress, the data in that backend will be available and writable although the index being rebuilt will not be used; therefore, searches which attempt to use that attribute might be unindexed.
- **Index Rebuild Administrative Alert**. The Directory Server generates an administrative alert when the rebuild process begins and ends. It will have a degraded-alert-type of "index-rebuild-in-progress" so that a Directory Proxy Server, such as the Directory Server can avoid using that server while the rebuild is in progress.
- **System Indexes Cannot be Rebuilt**. The contents of the backend must be exported and re-imported in order to rebuild system indexes. See the table below for the list of system indexes.
- **Indexing Certain Attributes**. You should ensure that the following recommendations are used when setting up the indexes.
  - Equality and substring indexes should not be used for attributes that contain binary data.
  - Approximate indexes should be avoided for attributes containing numbers, such as telephone numbers.
- **Unindexed Searches**. Unindexed attributes result in longer search times as the database itself has to be searched instead of the database index file. Only users with the `unindexed-search` privilege are allowed to carry out unindexed searches. In general, applications should be prevented from performing unindexed searches, so that

searches that are not indexed would be rejected rather than tying up a worker thread. The ways to achieve this include:

  - Make sure that only the absolute minimum set of users have the `unindexed-search` privilege. This privilege can be used without any other restrictions.
  - To allow unindexed searches with some control, the Permit Unindexed Search request control can be used with the `unindexed-search-with-control` privilege. With this privilege, a user will only be permitted to request an unindexed search if the search request includes the Permit Unindexed Search request control. The `unindexed-search privilege` allows a client to request an unindexed search without this control.
  - The Reject Unindexed Search request control can be used to explicitly indicate that a client does not want the server to process an unindexed search request, regardless of priviledges. See the LDAP SDK for information about these controls. These capabilities are also available with the `ldapsearch` tool.
  - Make sure that `allow-unindexed-searches` property is set to false in all client connection policies, in which unindexed searches should never be necessary. If the client connection policy should allow undindexed searches, set the `allow-unindexed-searches-with-control` property to true. If `allow-unindexed-searches` is false but, `allow-unindexed-searches-with-control` is true, the policy will only permit an unindexed search if the request includes the Permit Unindexed Search request control. See the LDAP SDK and the `ldapsearch` tool for more information.
  - Set a nonzero value for the `maximum-concurrent-unindexed-searches` global configuration property to ensure that if unindexed searches are allowed, only a limited number of them will be active at any given time. Administrators can configure the maximum number of concurrent unindexed searches by setting a property under Global Configuration.

    To change the maximum number of concurrent unindexed searches, use the `dsconfig` tool to set a value for the number. A value of "0" (default) represents no limit on the number of concurrent unindexed searches.

    ```
    $ bin/dsconfig set-global-configuration-prop \
      --set maximum-concurrent-unindexed-searches:2
    ```

- **Index Entry Limit**. The Directory Server specifies an index entry limit property. This property defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (default value is 4000), then you must rebuild the indexes using the `rebuild-index` tool. If an index entry limit value is set for the local DB backend, it overrides the value set for the overall JE backend index entry limit configuration (i.e., 4000).

  To change the default index entry limit, use the `dsconfig` tool as seen in the following example:

  ```
  $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
    --index-name cn --set index-entry-limit:5000
  ```

- **Rebuild Index vs Full Import**. You can expect a limited amount of database growth due to the existence of old data when running `rebuild-index` versus doing a full import of your database.

Index Types

The PingDirectory Server supports several types of indexes to quickly find entries that match search criteria in LDAP operations. The Directory Server uses an attribute's matching rules to normalize its values and uses those values as index keys to a list of matching entry IDs. Entry IDs are integer values that are used to uniquely identify an entry in the backend by means of a set of database index files (`id2entry`, `dn2id`, `dn2uri`, `id2children`, `id2subtree`).

Matching rules are elements defined in the schema that tell the server how to interact with the particular attribute. For example, the uid attribute has an equality matching rule defined in the schema, and thus, has an equality index maintained by the Directory Server. The following table describes the index types:

Table 14: Directory Server Index Types

Index Type	Description
Approximate	Used to efficiently locate entries that match the approximate search filter. It is used to identify which entries are approximately equal to a given assertion. Approximate indexes can only be applied to attributes that have a corresponding approximate matching rule.
Equality	Used to efficiently locate entries that match the equality search filter. It is used to identify which entries are exactly equal to a given assertion. Equality indexes can only be applied to attributes that have a corresponding equality matching rule. An offshoot of the equality index is the filtered index, which uses a defined search filter for a specific attribute. The filtered index can be maintained independently of the equality index for a specific attribute.
Ordering	Used to efficiently locate entries that match the ordering search filter. It is used to identify which entries have a relative order of values for an attribute. Ordering indexes can only be applied to attributes that have a corresponding ordering matching rule.
Presence	Used to efficiently locate entries that match the presence search filter. It is used to identify which entries have at least one value for a specified attribute. There is only one presence index key per attribute.
Substring	Used to efficiently locate entries that match the substring search filter. It is used to identify which entries contain specific substrings to a given assertion. Substring indexes can only be applied to attributes that have a corresponding substring matching rule.

System Indexes

The PingDirectory Server contains a set of system database index files that ensure that the server operates efficiently. These indexes cannot be modified or deleted.

Table 15: System Indexes

Index	Description
dn2id	Allows quick retrieval of DNs. The DN database, or `dn2id`, has one record for each entry. The key is the normalized entry DN and the value is the entry ID.
id2entry	Allows quick retrieval of entries. The `id2entry` database contains the LDAP entries. The database key is the entry ID and the value is the entry contents.
referral	Allows quick retrieval of referrals. The referral database called `dn2uri` contains URIs from referral entries. The key is the DN of the referral entry and the value is that of a labeled URI in the ref attribute for that entry.
id2children	Allows quick retrieval of an entry and its children. The `id2children` database provides a mapping between an entry's unique identifier and the entry unique identifiers of the corresponding entry's children.
id2subtree	Allows quick retrieval of an entry's subtree. The `id2subtree` database provides a mapping between an entry's unique identifier and its unique identifiers in its subtree.

To View the System Indexes

Use the `dbtest` command to view the system and user indexes. The index status indicates if it is a trusted index or not. An index is either trusted or untrusted. An untrusted index requires rebuilding.

```
$ bin/dbtest list-index-status --baseDN dc=example,dc=com --backendID
 userRoot
```

Managing Local DB Indexes

You can modify the local DB indexes to meet your system's requirements using the `dsconfig` tool. If you are using the `dsconfig` tool in interactive command-line mode, you can access the **Local DB Index** menu from the Basic object menu.

To View the List of Local DB Indexes

- Use `dsconfig` with the list-local-db-indexes option to view the default list of indexes.

```
$ bin/dsconfig list-local-db-indexes --backend-name userRoot
```

```
Local DB Index      : Type    : index-type
--------------------:---------:--------------------
aci                 : generic : presence
cn                  : generic : equality, substring
ds-entry-unique-id  : generic : equality
givenName           : generic : equality, substring
mail                : generic : equality
member              : generic : equality
objectClass         : generic : equality
sn                  : generic : equality, substring
telephoneNumber     : generic : equality
uid                 : generic : equality
uniqueMember        : generic : equality
```

To View a Property for All Local DB Indexes

- Use `dsconfig` with the --property option to view a property assigned set for all local DB indexes. Repeat the option for each property that you want to list. In this example, the `prime-index` property specifies if the backend is configured to prime the index at startup.

```
$ bin/dsconfig list-local-db-indexes --property index-entry-limit \
  --property prime-index --backend-name userRoot
```

To View the Configuration Parameters for Local DB Index

- To view the configuration setting of a local DB index, use `dsconfig` with the get-local-db-index-prop option and the `--index-name` and `--backend-name` properties. If you want to view the advanced properties, add the --advanced option to your command.

```
$ bin/dsconfig get-local-db-index-prop --index-name aci \
  --backend-name userRoot
```

To Modify the Configuration of a Local DB Index

You can easily modify an index using the `dsconfig` tool. Any modification or addition of an index requires the indexes to be rebuilt. In general, an index only needs to be built once after it has been added to the configuration.

If you add an index, then import the data using the `import-ldif` tool, then the index will be automatically rebuilt. If you add an index, then add the data using some other method than `import-ldif`, you must rebuild the index using the `rebuild-index` tool.

1. Use `dsconfig` with the set-local-db-index-prop option and the `--index-name` and `--backend-name` properties. In this example, update the `prime-index` property, which loads the index at startup. This command requires the --advanced option to access this property.

```
$ bin/dsconfig set-local-db-index-prop --index-name uid \
  --backend-name userRoot --set prime-index:true
```

2. View the index to verify the change.

```
$ bin/dsconfig get-local-db-index-prop --index-name uid \
  --backend-name userRoot
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in *General Tips on Indexes*.

```
$ bin/stop-server
```

4. Run the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index uid
```

5. Restart the Directory Server if shutdown.

```
$ bin/start-server
```

To Create a New Local DB Index

1. To create a new local DB index, use `dsconfig` with the --create-local-db-index option and the --index-name, --backend-name, and --set index-type: <value> options.

```
$ bin/dsconfig create-local-db-index \
  --index-name roomNumber --backend-name userRoot \
  --set index-type:equality
```

2. View the index.

```
$ bin/dsconfig get-local-db-index-prop \
  --index-name roomNumber --backend-name userRoot
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in *General Tips on Indexes* on page 148.

```
$ bin/stop-server
```

4. Rebuild the index using the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index roomNumber
```

5. Restart the Directory Server.

```
$ bin/start-server
```

To Delete a Local DB Index

You can delete an index using the `dsconfig` tool. Check that no plug-in applications are using the index before deleting it. When the index is deleted, the corresponding index database will also be deleted. The disk space is reclaimed once the cleaner threads begin.

- Use `dsconfig` with the `delete-local-db-index` option to remove it from the database.

```
$ bin/dsconfig delete-local-db-index \
  --index-name roomNumber --backend-name userRoot
```

Working with Composite Indexes

The PingDirectory Server composite index can be generated from multiple pieces of information (a combination of multiple filter components, or a combination of filter components and a base DN). A composite index can also be based on only a single piece of information.

To improve searches over a large number of entries, equality composite indexes can be used to combine a mandatory equality filter pattern with an optional base DN pattern to improve the performance of searches in directories with a very large number of entries, and in particular with a very large number of non-leaf entries. Equality composite indexes offer two advantages over existing equality attribute indexes in these types of deployments.

Base DN Pattern - If a directory environment has many branches, but searches are often done that are within specific individual branches, the base DN pattern can be used to make search processing more efficient. The server will only need to search entries within a target branch.

For example, if the directory contains an "`ou=Customers,dc=example,dc=com`" branch, with a separate branches below that for sets of customers, like "`ou=ACME,ou=Customers,dc=example,dc=com`", and "`ou=SHOPCO,ou=Customers,dc=example,dc=com`", a composite index with a filter pattern of "`(sn=?)`" and a base DN pattern of "`ou=?,ou=Customers,dc=example,dc=com`" can be defined. Then a search with a filter of "`(sn=Smith)`" and a base DN of "`ou=ACME,ou=Customers,dc=example,dc=com`" can be used to narrow the search to the Smiths in the ACME branch.

Index Pages - If many entries have the same value for a specific attribute, composite indexes can break large ID sets up across multiple pages, unlike the traditional attribute index. Using the previous example, if a search of the directory returns 50,000 Smiths, the results can be served in blocks of 5,000 IDs. An attribute index will return either one Smith record whose value is a block that contains all 50,000 of the matching entry IDs (if the index isn't exploded), or 50,000 Smith records that each have a value of the matching entry ID (when the index is exploded). The non-exploded form is efficient for searching because all of the entry IDs are returned in a single read, but it's expensive for writing because if a Smith must be added or removed, the entire block of 50,000 entry IDs must be rewritten. The exploded form is efficient for writing (adding or removing a Smith involves just that one entry ID), but it's expensive for searching because it takes 50,000 reads to get all entry IDs for all of the Smiths.

Composite indexes break up the block of entry IDs across multiple pages (a page size of up to 5000). If the directory contains 50,000 Smiths, instead of having to choose between one block of 50,000 IDs or 50,000 blocks of one ID, ten blocks of 5,000 IDs are returned. This improves the efficiency of a read or write across many entries.

There is little performance overhead to the paging mechanism. Use an equality composite index for an attribute that has a lot of entries that have the same value (such as givenName or sn), not for an attribute with very few entries with the same value (such as id or mail). For attributes in which all of the values match a small number of entries, it's better to use an equality attribute index.

When configuring a composite index, define the following properties:

Table 16: Composite Index Properties

Composite Index Properties	Description
index-filter-pattern	Specifies a single-valued filter property used to identify a portion of the index criteria. This can only be specified at the time that the index definition is created and is required.
index-base-dn-pattern	Specifies a single-valued DN property that may indicate that the index should be scoped to a specific subtree or subtree pattern. This can only be specified at the time the index definition is created and is optional.

Working with JSON Indexes

JSON indexing is similar to general attribute indexing. Where an attribute can be indexed several ways and requires a separate database for each index type, there is only a single database for each JSON field that can be used for different JSON filter types. This database primarily behaves like the database for an equality attribute index. Each database entry key is the normalized form for a value for the target JSON field, and the corresponding database entry value is an list of the entry IDs for all entries in which the associated attribute type has a JSON object with that value for the target field. The database is configured with a comparator (based on the data type for the target field) that enables iterating through values in a logical order to facilitate inequality and subInitial searches.

JSON indexes are automatically created when the JSON Field constraint indicates that a JSON field should be indexed. Indexes can be viewed with the following command:

```
$ bin/dbtest list-index-status \
  --backendID userRoot \
  --baseDN dc=example,dc=com
```

The JSON object filter types that can be enhanced through the use of JSON indexes include:

- `equals`. Identifies entries that have a specific value for the target field. This filter type only requires retrieving a single index key. However, depending on the nature of the search filter, the ID list may contain references to entries that don't actually match the filter (such as if the field is a string, and the filter is configured to use case-sensitive matching).
- `equalsAny`. Identifies entries that have any of a specified set of values for the target field. This filter type only requires retrieving the index keys that correspond to the target values in the filter and merging their ID lists.
- `greaterThan/lessThan`. Identifies entries that have at least one value for the target field that is greater or less than (or possibly equal to) a specified value. This index is similar in use to the `containsField` index, except that it only needs to iterate through a subset of the keys. A filter can contain both `greaterThan` and `lessThan` filters to represent a bounded range.
- `substring`. Identifies entries that have a string value for the target field that matches a given `substring`. The index can only be used for substring filters that include a `subInitial` component. In this case, the server iterates through all of the index keys that match the `startsWith` component, and manually compares values against the remainder of the substring assertion.

JSON indexing is available in local database backends backed by Berkeley DB Java Edition. This includes the following:

- Add, delete, modify, and modify DN operations that make changes to JSON objects stored in the server.
- LDIF imports that include JSON objects, including updates to the cache size estimates for the JSON indexes.
- The `rebuild-index` tool and corresponding backend code to make it possible to generate and rebuild indexes for JSON data. It must be possible to build all JSON indexes for all or a specified subset of fields associated with a given attribute type. The `verify-index` tool should also work with JSON indexes to make it possible to check their validity.
- Matching entry count control and debugsearchindex return attribute provide information about relevant JSON index usage.
- Support for monitoring index content and usage.

Note: Exploded indexes and the entry balancing global index do not support JSON objects.

Working with Local DB VLV Indexes

Local DB VLV indexes allow a client to request a subset of results from a sorted list that match a specific search base, scope, and filter. The client can navigate through the list by passing a context back to the server with the virtual list view control. The Local DB VLV index can be used only when the client request contains the virtual list view (VLV) control and the client has been authorized with an ACI with a `targetcontrol` of **2.16.840.1.113730.3.4.9**.

Note: A client request, which includes a virtual list view control, can be successfully processed without a matching Local DB VLV index if the search is completely indexed. This is not an efficient means of using VLV, since the server has to retrieve each entry twice.

To View the List of Local DB VLV Indexes

- Use `dsconfig` with the list-local-db-indexes option to view the default list of indexes. In the example, no VLV indexes are defined.

```
$ bin/dsconfig list-local-db-vlv-indexes --backend-name userRoot
```

To Create a New Local DB VLV Index

1. Use `dsconfig` with the create-local-db-vlv-index option and the --index-name, --backend-name, and --set index-type:(propertyValue) options. If you do not set any property values, the default values are assigned.

```
$ bin/dsconfig create-local-db-vlv-index \
  --index-name givenName --backend-name userRoot --set base-
dn:dc=example,dc=com \
  --set scope:whole-subtree --set filter:"(objectclass=*)" \
  --set sort-order:givenName
```

2. Rebuild the index using the `rebuild-index` tool. You must add the “vlv.” prefix to the index name to rebuild the VLV index. The following command can be run with the server on or offline with the addition of the `--task` and connection options.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

To Modify a VLV Index’s Configuration

1. Use `dsconfig` with the set-local-db-vlv-index-prop option and the `--index-name` and `--backend-name` properties. In this example, update the `base-dn` property.

```
$ bin/dsconfig set-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot --set base-dn:ou=People,dc=example,dc=com
```

2. Rebuild the index using the `rebuild-index` tool. You must add the prefix "vlv." to the index name. The following command can be run with the server on or offline with the addition of the `--task` and connection options.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

To Delete a VLV Index

You can delete a VLV index using the `dsconfig` tool. Check that the index is not being used in any plug-in applications before deleting it.

1. Use `dsconfig` with the delete-local-db-vlv-index option to remove it from the database.

```
$ bin/dsconfig delete-local-db-vlv-index --index-name givenName \
  --backend-name userRoot
```

2. Verify the deletion by trying to view the vlv index.

```
$ bin/dsconfig get-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot
```

Working with Filtered Indexes

The PingDirectory Server filtered index is useful when client search requests consisting of a compound &-filter with individual components matching a large number of entries, potentially greater than the index entry limit, have an intersection of a relatively small number of entries.

For example, assume a database contains several thousand company profiles and each company profile is represented by many entries. The "`(objectClass=company)`" filter matches a small set of entries per company, and therefore may exceed the index entry limit since there are many companies. Also assume that the "`(companyDomain=example.com)`" filter matches many of the entries for the company with domain `example.com` and can also result in an unindexed search. The more narrow filter "`(&(objectClass=company)(companyDomain=example.com))`" also results in an unindexed search but only matches a small number of entries. The filtered index makes it possible to index this compound filter by defining an equality index on the `companyDomain` attribute with a static filter of "`(objectClass=company)`" in the `equality-index-filter` property of the index.

Filtered indexing is primarily useful for cases in which clients frequently issue searches with AND filters that meet the following criteria:

- The AND filter itself matches a relatively small number of entries, but each of the individual components may match a very large number of entries.
- The filter has a dynamic component that does change, and that dynamic component always uses the same attribute.
- The filter has a static component that doesn't change.
- The filter must be narrowed to a base DN, for data structures with many branches, or if indexed attribute values appear in a very large number of entries.

The filtered index can be maintained independently from the equality filter for that attribute. Further, the filtered index will be used only for searches containing the equality component with the associated attribute type ANDed with this filter. When configuring a filtered index, be aware of the `equality-index-filter` and `maintain-equality-index-without-filter` properties of the index.

Once configured and built with the `rebuild-index` tool or `import-ldif`, searches with filters based on the above example will be processed with the index:

```
(&(objectClass=company)(companyDomain=example.com))
(&(objectClass=company)(|(companyDomain=example.com)
(companyDomain=example.org)))
(&(companyDomain=example.com)(objectClass=company))
(&(companyDomain=example.com)(&(objectClass=company)))
(&(companyDomain=example.com)(objectClass=company)(something=else))
(&(companyDomain=example.com)(&(objectClass=company)(something=else)))
(|(&(objectClass=company)(companyDomain=example.com))(&(objectClass=company)
(companyDomain=example.org)))
```

When configuring a filtered index, define the following properties:

Table 17: Filtered Index Properties

Filtered Index Properties	Description
equality-index-filter	Specifies a search filter that may be used in conjunction with an equality component for the associated attribute type. If an equality index filter is defined, then an additional equality index will be maintained for the associated attribute, but only for entries that match the provided filter. Further, the index will be used only for searches containing an equality component with the associated attribute type ANDed with this filter.
maintain-equality-index-without-filter	Specifies whether to maintain a separate equality index for the associated attribute without any filter, in addition to maintaining an index for each equality index filter that is defined. If this is false, then the attribute will not be indexed for equality by itself but only in conjunction with the defined equality index filters.

To Create a Filtered Index

1. Use the `dsconfig` tool to create a filtered index. The following command creates an equality index on the `companyDomain` attribute and maintains an index for the equality filter defined `"(objectclass=company)"`. After you have created the index, you must rebuild the indexes.

```
$ bin/dsconfig create-local-db-index --backend-name "userRoot" \
  --index-name companyDomain --set maintain-equality-index-without-
filter:true \
  --set index-type:equality --set equality-index-
filter:"(objectclass=company)"
```

2. Stop the Directory Server using `bin/stop-server`.
3. Run the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index companyDomain
```

4. Start the Directory Server using `bin/start-server`.

Tuning Indexes

The PingDirectory Server provides several tools to help you optimize your indexes and improve the overall read and write performance for your system. The server now supports an optional expanded index database using an *exploded* format to process high write load operations. To view the current state of the server's indexes and make adjustments to the index databases, the Directory Server automatically generates an Index Summary Statistics Table after each LDIF import or index rebuild. The `dbtest` tool also includes an Index Histogram to determine the key datasize for the indexes. This section provides descriptions of each of these tools.

About the Exploded Index Format

The Directory Server has two index properties on the Backend configuration object that can be used to optimize the server's performance: `index-entry-limit` and `exploded-index-entry-threshold`. The `index-entry-limit` specifies the maximum size of entries kept in an index record, before the server stops maintaining that record, begins scanning the whole database, and runs an expensive unindexed search. Typically, if any index keys have already reached this limit, indexes must be rebuilt before they can be allowed to use a new limit.

The `exploded-index-entry-threshold` property specifies the maximum number of entries that are allowed to match a given index key before that particular index key is stored in a separate database in an expanded (i.e., "exploded") format. An `exploded-index-entry-threshold` value of 0, or any value greater or equal to the `index-entry-limit`, means that the expanded index database will not be used. All keys whose entry count is less than `exploded-index-entry-threshold` continue to be stored in one database in a consolidated format, such that changes to the key require *rewriting* all the entry IDs matching the key. All keys whose entry count is greater than `exploded-index-entry-threshold` and less than the `index-entry-limit` are stored in a *separate* database in an exploded format, such that changes to the key require writing only to the updated entry ID.

For example, as shown in the figure below, the equality index for the `sn` attribute is stored in consolidated format, listing each entry ID for all entries that contain the `sn=smith` attribute. If we enable the `exploded-index-entry-threshold` property by setting its value to 1000 and keep the `index-entry-limit` property to its default limit of 4000, then the expanded index database will be implemented by the server. Any key with an entry count less than 1000 continues to be stored in consolidated format. If a key has an entry count greater than 1000, it will be stored in a separate database where each key is stored with its entry ID individually. The consolidated format is very efficient for read operations because the server can retrieve a row of entry IDs at once, while the exploded format is far more efficient for write operations.

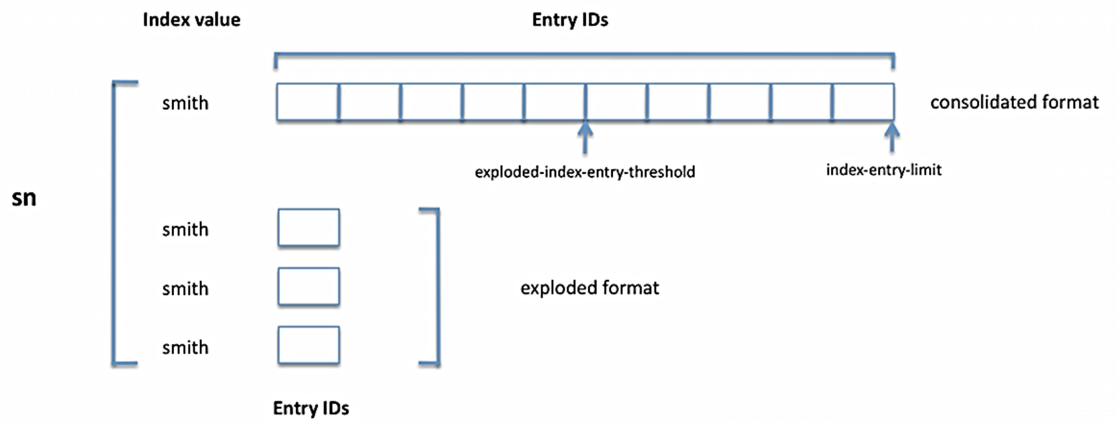


Figure 2: Consolidated and Exploded Index Formats

Note that a large value for the `exploded-index-entry-threshold` can have a big impact on write performance and database growth on disk, since each change to an index key stored in the consolidated format requires that IDs for all entries matching the key be rewritten to the database. For example if the exploded-index-entry-threshold is set to 100,000, and the number of entries matching a given key is 50,000, then changes to that key will require writing 200KB to the database since each entry ID is 4 bytes. Add and delete operations are especially impacted since they must update all indexes for an entry. If many indexes have a large exploded-index-entry-threshold, this can lead to very low throughput and can cause the database to temporarily grow very large on disk. It is recommended to set this value at or below 10,000, and never set it above 50,000. If any index keys have already reached this threshold, indexes need to be rebuilt before they are allowed to use the new threshold. This is also true for the index-entry-limit property: setting a large limit (greater than 10,000) could have a big impact on write performance and database growth on disk.

Monitoring Index Entry Limits

Index keys that have reached their limit require indexes to be rebuilt before they can be allowed to use a new limit. There are several ways to monitor index limits to avoid a potentially costly rebuild. There are cases in which it is acceptable for index keys to exceed the index entry limit. For example, the objectClass attribute type should be indexed for equality because the server needs to use it to find all group entries when bringing a backend online, and also because applications frequently need to find entries of a specific type. However, it doesn't make sense for the "top" objectClass key to be indexed, because it appears in every entry in the server.

Choose an index entry limit value that is high enough to ensure that all of the right keys are indexed, but keys that occur too frequently are not. The verify-index tool --listKeysNearestIndexEntryLimt argument lists a specified number of keys that are closest to the limit without having exceeded it. The index entry limit should be larger than the number of entries matching the largest key to remain indexed, with enough overhead to account for future growth. Use this command regularly to determine if the index entry limit needs to be adjusted.

The verify-index tool also provides the --listKeysExceedingIndexEntryLimit argument to list all keys for which the value has exceeded the index entry limit and the number of entries in which they appear. If there are keys for which the limit has already exceeded but that need to be maintained, adjust the index entry limit to be higher than the number of entries that contain that key (with additional room for future growth) and run the rebuild-index tool (or export to LDIF and re-import).

The server provides other ways to determine if index keys have exceeded, or are close to exceeding, the index entry limit. Some of these include:

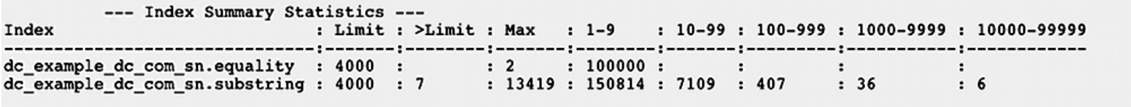
- When performing an LDIF import, the tool includes an "Index Summary Statistics" section that provides usage information for each index, including the number of keys for which the index entry limit has been exceeded, and also the number of keys for which the number of matching entries falls within a number of predefined buckets (such as 1–4 entries, 5–9 entries, 10–99 entries, and 100–999 entries).
- If, during a search operation, the server accesses one or more index keys whose values have exceeded the index entry limit, the access log message for that operation will include an indexesWithKeysAccessedExceedingEntryLimit field containing a comma-delimited list of the appropriate indexes. The same access log field may appear in log messages for add, delete, modify, and modify DN operations in which the server wrote to, or tried to write to, at least one index key whose value exceeded the index entry limit.
- If, during a search operation, the server accesses one or more index keys whose values have not yet exceeded the index entry limit but are more than 80 percent to reaching that limit, the access log message for that operation will include an indexesWithKeysAccessedNearEntryLimit field containing a comma-delimited list of the appropriate indexes. The same access log field may appear in log messages for add, delete, modify, and modify DN operations in which the server wrote to at least one index key whose value was within 80 percent of the index entry limit.
- If a search operation requests either includes the debugsearchindex attribute, or the matching entry count request control with debugging enabled, the debug information will include any indexes accessed that have exceeded the index entry limit, or that are within 80 percent of the configured index entry limit.
- The monitor entry for each configured index includes attributes that provide information about the number of index keys that have been encountered (since the backend was brought online, or since the index entry limit was changed) in a number of different categories. These monitor attributes include:

- `ds-index-exceeded-entry-limit-count-since-db-open` — The number of index keys for which the number of matching entries has crossed the index entry limit due to a write operation.
- `ds-index-unique-keys-near-entry-limit-accessed-by-search-since-db-open` — The number of unique index keys that have been accessed by a search operation for which the number of matching entries is within 80 percent of the index entry limit.
- `ds-index-unique-keys-exceeding-entry-limit-accessed-by-search-since-db-open` — The number of unique index keys that have been accessed by a search operation for which the number of matching entries has exceeded the index entry limit at some point since the index was last built.
- `ds-index-unique-keys-near-entry-limit-accessed-by-write-since-db-open` — The number of unique index keys that have been accessed by a write operation for which the number of matching entries is within 80 percent of the index entry limit.
- `ds-index-unique-keys-exceeding-entry-limit-accessed-by-write-since-db-open` — The number of unique index keys that have been accessed by a write operation for which the number of matching entries has exceeded the index entry limit at some point since the index was last built.

About the Index Summary Statistics Table

The Directory Server now automatically generates an Index Summary Statistics Table, which can be used to determine the optimal configuration for your system's indexes. This table is only generated when the `rebuild-index` tool is run in offline mode. The table is generated after any LDIF import or an index rebuild, and is written to system out and to `logs/tools/rebuild-index-summary.txt`. The table lists the current index entry limit (set by the `index-entry-limit` property on the local DB configuration), the number of keys whose entry count exceeds this limit if any, and the maximum entry count for any key in the index. The table then displays a histogram of the number of keys whose entry falls within a range of values. An example of the Index Summary Statistics table is show below for the `sn.equality` and the `sn.substring` indexes.

The following figure shows that there are seven substrings whose entry counts exceed the `index-entry-limit` of 4000. Six of the substrings are in the 10000-99999 range with the maximum entry count being 13419. By deduction, one more substring must be present in the 1000-9999 range that exceeds the `index-entry-limit` of 4000. These substrings could be expensive for search operations.

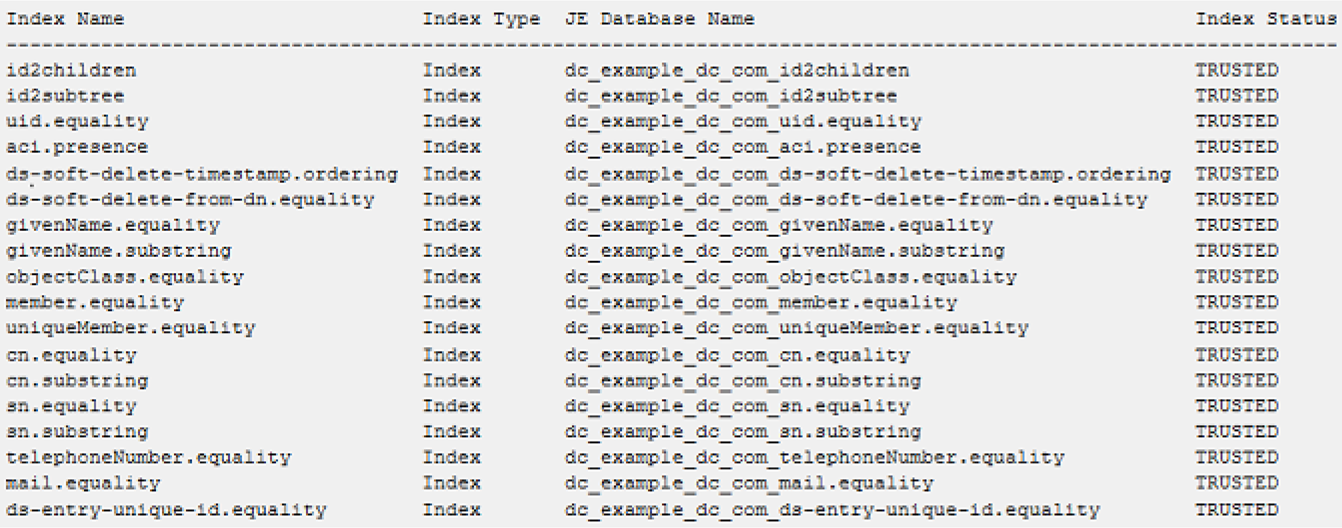


```
          --- Index Summary Statistics ---
Index                          : Limit : >Limit : Max   : 1-9    : 10-99 : 100-999 : 1000-9999 : 10000-99999
-------------------------------:-------:--------:-------:--------:-------:---------:-----------:------------
dc_example_dc_com_sn.equality  : 4000  :        : 2     : 100000 :       :         :           :
dc_example_dc_com_sn.substring : 4000  : 7      : 13419 : 150814 : 7109  : 407     : 36        : 6
```

Figure 3: Example of an Index Summary Statistics Table

About the dbtest Index Status Table

The `dbtest` tool has a `list-all --analyze` option that generates the current status of all of the databases on your system, including all index databases. The table shows the type, entry count (i.e., the number of records in the dtabase), index status (TRUSTED to indicate that the indexes are up-to-date, or UNTRUSTED if the index needs rebuilding), the total data size for each key, the average data size for each key and the maximum data size for each key. Note also that any indexes that are in exploded format are listed on this table.



```
Index Name                         Index Type  JE Database Name                                  Index Status
-------------------------------------------------------------------------------------------------------------
id2children                        Index       dc_example_dc_com_id2children                     TRUSTED
id2subtree                         Index       dc_example_dc_com_id2subtree                      TRUSTED
uid.equality                       Index       dc_example_dc_com_uid.equality                    TRUSTED
aci.presence                       Index       dc_example_dc_com_aci.presence                    TRUSTED
ds-soft-delete-timestamp.ordering  Index       dc_example_dc_com_ds-soft-delete-timestamp.ordering  TRUSTED
ds-soft-delete-from-dn.equality    Index       dc_example_dc_com_ds-soft-delete-from-dn.equality    TRUSTED
givenName.equality                 Index       dc_example_dc_com_givenName.equality              TRUSTED
givenName.substring                Index       dc_example_dc_com_givenName.substring             TRUSTED
objectClass.equality               Index       dc_example_dc_com_objectClass.equality            TRUSTED
member.equality                    Index       dc_example_dc_com_member.equality                 TRUSTED
uniqueMember.equality              Index       dc_example_dc_com_uniqueMember.equality           TRUSTED
cn.equality                        Index       dc_example_dc_com_cn.equality                     TRUSTED
cn.substring                       Index       dc_example_dc_com_cn.substring                    TRUSTED
sn.equality                        Index       dc_example_dc_com_sn.equality                     TRUSTED
sn.substring                       Index       dc_example_dc_com_sn.substring                    TRUSTED
telephoneNumber.equality           Index       dc_example_dc_com_telephoneNumber.equality        TRUSTED
mail.equality                      Index       dc_example_dc_com_mail.equality                   TRUSTED
ds-entry-unique-id.equality        Index       dc_example_dc_com_ds-entry-unique-id.equality     TRUSTED
```

Figure 4: dbtest Output Including Index Databases

Configuring the Index Properties

By default, the `index-entry-limit` and the `exploded-index-entry-threshold` are set to 4000, which means that the latter property is not enabled due to its having an equal value to that of the `index-entry-limit` property. You can enable the exploded index database by changing the `exploded-index-entry-threshold` property to a value less than the `index-entry-limit` and to a non-zero value using the `dsconfig` tool.

To Configure the Index Properties

Before running the following commands, be aware that you will need to do an index rebuild on the system, which requires a system shutdown, unless the command is run as a task.

1. Run `dsconfig` and set the `index-entry-limit` and the `exploded-index-entry-threshold` to 5000 and 1000, respectively. By default, both values are set to 4000. Changing the `exploded-index-entry-threshold` property to a value less than the `index-entry-limit` and to a non-zero value enables the exploded index database. Remember to include the bind parameters for your system. Once you enter the command, you must confirm that you want to apply the changes to your system.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set index-entry-limit:5000 \
  --set exploded-index-entry-threshold:1000
```

```
One or more configuration property changes require administrative action
or confirmation/notification.  Those properties include:

 * index-entry-limit:  If any index keys have already reached this limit,
 indexes
   must be rebuilt before they will be allowed to use the new limit.
   Setting a large limit (greater than 10,000) could have a big impact on
   write performance and database growth on disk.
 * exploded-index-entry-threshold:  If any index keys have already reached
   this threshold, indexes need to be rebuilt before they are allowed to use
 the
   new threshold. Setting a large limit (greater than 10,000) could have a
 big
   impact on write performance and database growth on disk.

Continue?  Choose 'no' to return to the previous step (yes / no) [yes]: yes
```

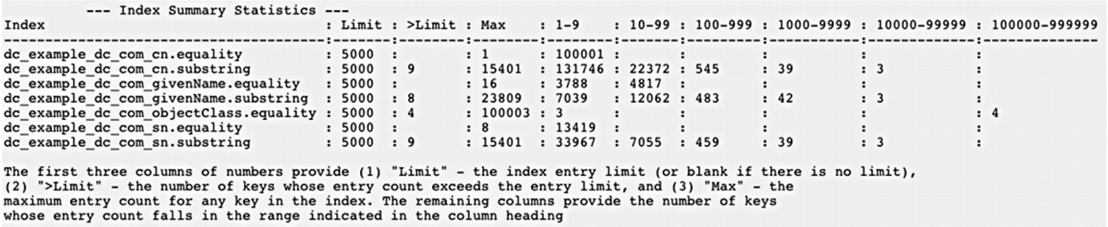
2. Stop the server.

```
$ bin/stop-server
```

3. Rebuild the index.

```
$ bin/rebuild-index --baseDN dc=example,dc=com \
  --index cn --index givenName --index objectClass \
  --index sn --maxThreads 10
```

4. View the Index Summary Statistics table, which is automatically displayed to system out after running the `rebuild-index` command. You can also access the table at `logs/tools/rebuild-index-summary.txt`. Repeat the last three steps to make more adjustments to your indexes.



```
          --- Index Summary Statistics ---
Index                                  : Limit : >Limit : Max    : 1-9    : 10-99 : 100-999 : 1000-9999 : 10000-99999 : 100000-999999
---------------------------------------:-------:--------:--------:--------:-------:---------:-----------:-------------:--------------
dc_example_dc_com_cn.equality          : 5000  :        : 1      : 100001 :       :         :           :             :
dc_example_dc_com_cn.substring         : 5000  : 9      : 15401  : 131746 : 22372 : 545     : 39        : 3           :
dc_example_dc_com_givenName.equality   : 5000  :        : 16     : 3788   : 4817  :         :           :             :
dc_example_dc_com_givenName.substring  : 5000  : 8      : 23809  : 7039   : 12062 : 483     : 42        : 3           :
dc_example_dc_com_objectClass.equality : 5000  : 4      : 100003 : 3      :       :         :           :             : 4
dc_example_dc_com_sn.equality          : 5000  :        : 8      : 13419  :       :         :           :             :
dc_example_dc_com_sn.substring         : 5000  : 9      : 15401  : 33967  : 7055  : 459     : 39        : 3           :

The first three columns of numbers provide (1) "Limit" - the index entry limit (or blank if there is no limit),
(2) ">Limit" - the number of keys whose entry count exceeds the entry limit, and (3) "Max" - the
maximum entry count for any key in the index. The remaining columns provide the number of keys
whose entry count falls in the range indicated in the column heading
```

5. Restart the server.

```
$ bin/start-server
```

Chapter 11

Managing Entries

Topics:

The Directory Server is a fully LDAPv3-compliant server that comes with a comprehensive set of LDAP command-line tools to search, add, modify, and delete entries.

This chapter presents the following topics:

Searching Entries

The Directory Server provides an ldapsearch tool to search for entries or attributes within your server. The tool requires the LDAP connection parameters needed to bind to the server, including the baseDN option to specify the starting point of the search within the server, and the search scope. The searchScope option determines the depth of the search:

base (search only the entry specified)
one (search only the children of the entry and not the entry itself)
sub (search the entry and its descendents)

The ldapsearch tool provides basic functionality as specified by the RFC 2254 but provides additional features that takes advantage of the Directory Server's control mechanisms. For more information, run the ldapsearch --help function.

To Search the Root DSE

The Root DSE is a special entry that resides at the root of the directory information tree (DIT). The entry holds operational information about the server and its supported controls. Specifically, the root DSE entry provides information about the supported LDAPv3 controls, SASL mechanisms, password authentication schemes, supported LDAP protocols, additional features, naming contexts, extended operations, and server information.

Note: The Directory Server provides an option to retrieve the Root DSE's operational attributes and add them to the user attribute map of the generated entry. This feature allows client applications that have difficulty handling operational attributes to access the root DSE using the show-all-attributes configuration property. Once this property is set, the associated attribute types are re-created and re-registered as user attributes in the schema (in memory, not on disk). Once you set the property, you can use ldapsearch without "+" to view the root DSE.

Use the dsconfig tool to set the show-all-attributes property to TRUE, as follows:

```
$ bin/dsconfig set-root-dse-backend-prop --set show-all-attributes:true
```

- Use ldapsearch to view the root DSE entry on the Directory Server. Be sure you include the "+" to display the operational attributes in the entry.

```
$ bin/ldapsearch --baseDN "" --searchScope base "(objectclass=*)" "+"
```

To Search All Entries in the Directory Server

- Use ldapsearch to search all entries in the Directory Server. The filter "(objectclass=*)" matches all entries. If the --searchScope option is not specified, the command defaults to a search scope of sub:

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --searchScope sub "(objectclass=*)"
```

To Search for an Access Control Instruction

- Use ldapsearch to search the dc=example,dc=com base DN entry. The filter "(aci=*)" matches all aci attributes under the base DN, and the aci attribute is specified so that only it is returned. The cn=Directory Manager bind DN has the privileges to view an ACI.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(aci=*)" aci
```

```
dn: dc=example,dc=com
aci: (targetattr!="userPassword")
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Search for the Schema

- Use `ldapsearch` to search the `cn=schema` entry. The base DN is specified as `cn=schema`, and the filter `"(objectclass=*)"` matches all entries. The command uses a special attribute "+" to return all operational attributes:

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base "(objectclass=*)" "+"
```

To Search for a Single Entry using Base Scope and Base DN

- Use `ldapsearch` to search for a single entry by specifying the base scope and DN:

```
$ bin/ldapsearch --baseDN uid=user.14,ou=People,dc=example,dc=com \
  --searchScope base "(objectclass=*)"
```

To Search for a Single Entry Using the Search Filter

- Search for a single entry by specifying the sub scope and a search filter that describes a single entry:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub "(uid=user.14)"
```

To Search for All Immediate Children for Restricted Return Values

- Search for all immediate children of `ou=People,dc=example,dc=com`. The attributes returned are restricted to `sn` and `givenName`. The special attribute "+" returns all operational attributes:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope one '(objectclass=*)' sn givenName "+"
```

To Search for All Children of an Entry in Sorted Order

- Search for all children of the `ou=People,dc=example,dc=com` subtree. The resulting entries are sorted by the server in ascending order by `sn` and then in descending order by `givenName`:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub --sortOrder sn,-givenName '(objectclass=*)'
```

To Limit the Number of Returned Search Entries and Search Time

- Search for a subset of the entries in the `ou=People,dc=example,dc=com` subtree by specifying a compound filter. No more than 200 entries will be returned and the server will spend no more than 5 seconds processing the request. Returned attributes are restricted to a few operational attributes:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub --sizeLimit 200 --timeLimit 5 \
  "(&(sn<=Doe)(employeeNumber<=1000))" ds-entry-unique-id entryUUID
```

To Get Information about How Indexes are used in a Search Operation

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Information about the can be obtained about the server's use of indexes in the course of processing a search operation. The following are two basic ways to accomplish this.

Issue a search request with the desired base DN, scope, and filter, and request that the server return the special `debugsearchindex` attribute. Users will need to be granted access to the `debugsearchindex` operational attribute and the `cn=debugsearch` portion of the DIT with the following command:

```
$ bin/dsconfig set-access-control-handler-prop \
 --add "global-aci:(targetattr=\"debugsearchindex\")(target=\"ldap:///
cn=debugsearch\")
    (version 3.0; acl \"Allow members of the Index Debugging Users group
    to request the debugsearchindex operational attribute \"; allow
    (read,search,compare) groupdn=\"ldap:///cn=Index Debugging
    Users,ou=Groups,dc=example,dc=com\";)"
```

To issue a search request for the server return the special `debugsearchindex` attribute, use the `ldapsearch` command such as:

```
$ bin/ldapsearch --hostname ds.example.com \
 --port 389 --bindDN uid=admin,dc=example,dc=com \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --searchScope sub "(&(givenName=John)(sn=Doe))" debugsearchindex
 dn: cn=debugsearch
 debugsearchindex: 0.040 ms - Beginning index processing for search
     request with base DN 'dc=example,dc=com', scope wholeSubtree,
     and filter (&(givenName=John)(sn=Doe)).
 debugsearchindex: 0.067 ms - Unable to optimize the AND filter
     beyond what the client already provided.
 debugsearchindex: 0.834 ms - Candidate set obtained for single-key
     filter (givenName=John) from index dc_example_dc_com_givenName.equality.
     Candidate set:  CandidateSet(isDefined=true, isExploded=false,
     isResolved=true, size=2, originalFilter=(givenName=John),
     remainingFilter=null, matchingEntryCountType=UNEXAMINED_COUNT)
 debugsearchindex: 0.030 ms - Final candidate set for filter (givenName=John)
     obtained from an unexploded index key in
 dc_example_dc_com_givenName.equality.
     Since the scope of the search includes the entire entry container, there
 is
     no need to attempt to further pare down the results based on the search
 scope.
     Candidate set:  CandidateSet(isDefined=true, isExploded=false,
 isResolved=true,
     size=2, originalFilter=(givenName=John), remainingFilter=null,
     matchingEntryCountType=UNEXAMINED_COUNT)
 debugsearchindex: 0.020 ms - Short-circuiting index processing for AND filter
     (&(givenName=John)(sn=Doe)) after evaluating single-key component
     (givenName=John) because the current ID set size of 2 is within the
      short-circuit threshold of 5.
 debugsearchindex: 0.030 ms - Obtained a candidate set of size 2 for AND
 filter
     (&(givenName=John)(sn=Doe)) with remaining filter (sn=Doe).  Even though
     there is still more of the filter to evaluate, the current candidate set
 is
     within the short-circuit threshold of 5, so no additional index
 processing will
     be performed to try to pare down the results based on the remaining
 filter or the
     search scope.  Candidate set:  CandidateSet(isDefined=true,
 isExploded=false,
     isResolved=true, size=2, originalFilter=(&(givenName=John)(sn=Doe)),
     remainingFilter=(sn=Doe), matchingEntryCountType=UPPER_BOUND)
 debugsearchindex: 0.016 ms - Completed all index processing.  Candidate set:
     CandidateSet(isDefined=true, isExploded=false, isResolved=true, size=2,
     originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
     matchingEntryCountType=UPPER_BOUND)
```

The second way would be to issue a search request with the desired base DN, scope, and filter, and include the matching entry count request control with the debug option set to `true`. To do this with `ldapsearch`, use a command such as:

```
$ bin/ldapsearch --hostname ds.example.com --port 389 \
 --bindDN uid=admin,dc=example,dc=com --bindPassword password \
 --baseDN dc=example,dc=com \
 --searchScope sub --matchingEntryCountControl examineCount=0:debug
 "(&(givenName=John)(sn=Doe))"
 Upper Bound on Matching Entry Count:  2
 Matching Entry Count Debug Messages:
 * naw-desktop:1389 - 0.104 ms - Beginning index processing for search request
 with
    base DN 'dc=example,dc=com', scope wholeSubtree, and filter
 (&(givenName=John)(sn=Doe)).
 * naw-desktop:1389 - 0.105 ms - Unable to optimize the AND filter beyond what
 the client already provided.
 * naw-desktop:1389 - 0.614 ms - Candidate set obtained for single-key filter
 (givenName=John) from index
    dc_example_dc_com_givenName.equality.  Candidate set:
    CandidateSet(isDefined=true, isExploded=false, isResolved=true,
    size=2, originalFilter=(givenName=John), remainingFilter=null,
 matchingEntryCountType=UNEXAMINED_COUNT)
 * naw-desktop:1389 - 0.090 ms - Final candidate set for filter
    (givenName=John) obtained from an unexploded index key in
    dc_example_dc_com_givenName.equality.  Since the scope of the search
    includes the entire entry container, there is no need
    to attempt to further pare down the results based on the search scope.
    Candidate set:  CandidateSet(isDefined=true, isExploded=false,
 isResolved=true,
    size=2, originalFilter=(givenName=John), remainingFilter=null,
 matchingEntryCountType=UNEXAMINED_COUNT)
 * naw-desktop:1389 - 0.045 ms - Short-circuiting index processing for AND
 filter
    (&(givenName=John)(sn=Doe)) after evaluating single-key component
    (givenName=John) because the current ID set size of 2 is within the short-
circuit threshold of 5.
 * naw-desktop:1389 - 0.111 ms - Obtained a candidate set of size 2 for AND
 filter
    (&(givenName=John)(sn=Doe)) with remaining filter (sn=Doe).  Even though
 there is
    still more of the filter to evaluate, the current candidate set is within
    the short-circuit threshold of 5, so no additional index processing will
 be performed
    to try to pare down the results based on the remaining filter or the
 search scope.
    Candidate set:  CandidateSet(isDefined=true, isExploded=false,
 isResolved=true, size=2,
    originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
    matchingEntryCountType=UPPER_BOUND)
 * naw-desktop:1389 - 0.040 ms - Completed all index processing.  Candidate
 set:
    CandidateSet(isDefined=true, isExploded=false, isResolved=true, size=2,
    originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
    matchingEntryCountType=UPPER_BOUND)
 * naw-desktop:1389 - The search is partially indexed
 (candidatesAreInScope=true,
    unindexedFilterPortion=(sn=Doe))
 * naw-desktop:1389 - Constructing an UPPER_BOUND response with a count of 2
```

Working with the Matching Entry Count Control

The ldapsearch command can be used with the --matchingEntryCountControl option to determine the count of entries that match a search filter. Users will need to be granted access to this control by its OID 1.3.6.1.4.1.30221.2.5.36 with the following command:

```
$ bin/dsconfig set-access-control-handler-prop \
 --add "global-aci:(targetcontrol=\"1.3.6.1.4.1.30221.2.5.36\")
   (version 3.0; acl \"Allow members of the Index Debugging Users group to
   use the matching entry count request control \"; allow (read)
   groupdn=\"ldap:///cn=Index Debugging Users,ou=Groups,dc=example,dc=com\";)"
```

An examineCount control can be used for searches that are at least partially indexed, in order to determine whether to return an examined count, an unexamined count, or an upper bound count. The factors that determine what is returned are:

- A search is fully indexed if indexes can be used to identify the entry IDs for all entries that match the filter without ambiguity. Indexes can also be used to make sure that all of those candidates are within the scope of the search.
- A search is partially indexed if indexes can be used to identify the entry IDs for all entries that match the search criteria, but the candidate list may potentially also include entries that either don't match the filter or are outside the scope of the search.
- A search is unindexed if it is not possible to retrieve a candidate list based on either the filter or the search scope.
- An unexamined count is a count of the exact number of entries that match the search criteria, only through the use of index processing.
- An examined count is the same as an unexamined count, except that all of the candidate entries are examined to determine whether they would have been returned to the client. An examined count may be less than an unexamined count if the set of matching entries includes those that would be removed by access control evaluation, or special entries like LDAP sub-entries, replication conflict entries, or soft-deleted entries.
- An upper bound count is the maximum number of entries that match the criteria, but indicates that the server could not determine exactly how many matching entries there were without examining each candidate, which it did not do.
- If a search is fully indexed, the result is an examined count or an unexamined count. If alwaysExamine is true and examineCount is greater than or equal to the number of candidates, the result is an examined count. If alwaysExamine is false, or if the number of candidates exceeds examineCount, the result is an unexamined count.
- If a search is partially indexed, the result is either an examined count or an upper bound count. The alwaysExamine flag isn't relevant in this case. If examineCount is greater than or equal to the number of candidates, the result is an examined count. If not, the result is an upper bound count.
- If a search is unindexed, the result is either an examined count or an unknown count. If allowUnindexed is true, the unindexed search is processed, which can be very expensive. Instead of getting the matching entries back, the examined count is returned. If allowUnindexed is false, an unknown count is returned. If allowUnindexed is true, the requester needs to have the unindexed-search privilege to get the exact count.

The following is a sample ldapsearch with the --matchingEntryCountControl option:

```
$./ldapsearch \
  --bindDN "cn=directory manager" \
  --bindPassword password \
  --baseDN dc=example,dc=com \
  --matchingEntryCountControl
 examineCount=100:alwaysExamine:allowUnindexed:debug \
  "(objectclass=*)"
```

Adding Entries

Depending on the number of entries that you want to add to your Directory Server, you can use the `ldapmodify` tool for small additions. The `ldapmodify` tool provides two methods for adding a single entry: using a LDIF file or from the command line. The attributes must conform to your schema and contain the required object classes.

Adding requests with the `ignore-no-user-modification` control enable a client to include attributes that are not normally allowed from external sources. For example, the `userPassword` attribute is a user-modifiable attribute. An add request with the `ignore-no-user-modification` control allows a one-time exception to the password policy, even if the requesting client does not have the `bypass-pw-policy` privilege. This exception enables specifying pre-encoded passwords.

Note: When adding an entry, the server can ensure that the entry's RDN is unique and does not contain any sensitive information by replacing the provided entry's RDN with the server-generated entryUUID value. An LDAP client written with the LDAP SDK for Java can use the `NameWithEntryUUIDRequestControl` to explicitly indicate which add requests should be named in this way, or the `ldapmodify` tool with the `--nameWithEntryUUID` argument. Also, the `auto-name-with-entry-uuid-connection-criteria` and `auto-name-with-entry-uuid-request-criteria` global configuration properties can be used to identify which add requests should be automatically named this way.

The uniqueness request control can also be used with `ldapmodify` for enforcing uniqueness on a per-request basis. Provide at least one of the `uniquenessAttribute` or `uniquenessFilter` arguments with the request. For more information about this control, see the LDAP SDK documentation and the `com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl` class for using the control.

To Add an Entry Using an LDIF File

1. Open a text editor and create an entry that conforms with your schema. For example, add the following entry in the file and save the file as `add-user.ldif`. For the `userPassword` attribute, enter the cleartext password. The Directory Server encrypts the password and stores its encrypted value in the server. Make sure that the LDIF file has limited read permissions for only authorized administrators.

```
dn: uid=user.2000,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Toby Hall$73600 Mash Street$Cincinnati, OH 50563 postalCode:
 50563
description: This is the description for Toby Hall.
uid: user.2000
userPassword: wordsmith employeeNumber: 2000
initials: TBH
givenName: Toby
pager: +1 596 232 3321
mobile: +1 039 311 9878
cn: Toby Hall
sn: Hall
telephoneNumber: +1 097 678 9688
street: 73600 Mash Street
homePhone: +1 214 233 8484
l: Cincinnati
mail: user.2000@maildomain.net
st: OH
```

2. Use the `ldapmodify` tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. If the command is successful, you will see generated success messages with the "#" symbol.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

```
# Processing ADD request for uid=user.2000,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2000,ou=People,dc=example,dc=com
```

To Add an Entry Using the Changetype LDIF Directive

RFC 2849 specifies LDIF directives that can be used within your LDIF files. The most commonly used directive is `changetype`, which follows the `dn:` directive and defines the operation on the entry. The main advantage of using this method in an LDIF file is that you can combine adds and modifies in one file.

1. Open a text editor and create an entry that conforms with your schema. For example, add the following entry in the file and save the file as `add-user2.ldif`. Note the use of the `changetype` directive in the second line.

```
dn: uid=user.2001,ou=People,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Seely Dorm$100 Apple Street$Cincinnati, OH 50563
postalCode: 50563
description: This is the description for Seely Dorm.
uid: user.2001
userPassword: pleasantry
employeeNumber: 2001
initials: SPD
givenName: Seely pager: +1 596 665 3344
mobile: +1 039 686 4949
cn: Seely Dorm
sn: Dorm
telephoneNumber: +1 097 257 7542
street: 100 Apple Street
homePhone: +1 214 521 4883
l: Cincinnati
mail: user.2001@maildomain.net
st: OH
```

2. Use the `ldapmodify` tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. In this example, you do not need to use the --defaultAdd or its shortform -a option with the command.

```
$ bin/ldapmodify --filename add-user2.ldif
```

To Add Multiple Entries in a Single File

You can have multiple entries in your LDIF file by simply separating each DN and its entry with a blank line from the next entry.

1. Open a text editor and create a couple of entries that conform to your schema. For example, add the following entries in the file and save the file as `add-user3.ldif`. Separate each entry with a blank line.

```
dn: uid=user.2003,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass:
organizationalPerson
objectClass: inetOrgPerson
...(similar attributes to previous examples)...

dn: uid=user.2004,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

```
...(similar attributes to previous examples)...
```

2. Use the `ldapmodify` tool to add the entries specified in the LDIF file. In this example, we use the short form arguments for the `ldapmodify` tool.

   The -h option specifies the host name, the -p option specifies the LDAP listener port, -D specifies the bind DN, -w specifies the bind DN password, -a specifies that entries that omit a changetype will be treated as add operations, and -f specifies the path to the input file. If the operation is successful, you will see commented messages (those begining with "#") for each addition.

   ```
   $ bin/ldapmodify -h server.example.com -p 389 \
     -D "cn=admin,dc=example,dc=com" -w password -a -f add-user3.ldif
   ```

   ```
   # Processing ADD request for uid=user.2003,ou=People,dc=example,dc=com
   # ADD operation successful for DN uid=user.2003,ou=People,dc=example,dc=com
   # Processing ADD request for uid=user.2004,ou=People,dc=example,dc=com
   # ADD operation successful for DN uid=user.2004,ou=People,dc=example,dc=com
   ```

Deleting Entries Using ldapdelete

You can delete an entry using the `ldapdelete` tool. You should ensure that there are no child entries below the entry as that could create an orphaned entry. Also, make sure that you have properly backed up your system prior to removing any entries.

To Delete an Entry Using ldapdelete

Use `ldapdelete` to delete an entry. The following example deletes the `uid=user.14` entry.

```
$ bin/ldapdelete uid=user.14,ou=People,dc=example,dc=com
```

To Delete Multiple Entries Using an LDIF File

1. You can generate a file of DNs that you would like to delete from the Directory Server.

   The following command searches for all entries in the `ou=Accounting` branch and returns the DNs of the subentries.

   ```
   $ bin/dump-dns -D "cn=admin,dc=example,dc=com" -w password --baseDN \
                           "ou=Accounting,ou=People,dc=example,dc=com" --
   outputFile /usr/local/entry_dns.txt
   ```

2. Run the `ldapdelete` command with the file to delete the entries. The command uses the --continueError option, which will continue deleting through the whole list even if an error is encountered for a DN entry.

   ```
   $ bin/ldapdelete --filename /usr/local/entry_dns.txt --continueError
   ```

Deleting Entries Using ldapmodify

You can use the LDIF changetype directive to delete an entry from the Directory Server using the `ldapmodify` tool. You can only delete leaf entries.

To Delete an Entry Using ldapmodify

- From the command line, use the `ldapmodify` tool with the changetype:delete directive. Enter the DN, press **Enter** to go to the next line, then enter the changetype directive. Press **Control-D** twice to enter the EOF sequence (UNIX) or **Control-Z** (Windows).

  ```
  $ bin/ldapmodify --hostname server1.example.com -port 389 --bindDN
   "uid=admin,dc=example,dc=com" --bindPassword password
  dn:uid=user.14,ou=People,dc=example,dc=com
  ```

```
changetype: delete
```

Modifying Entries Using ldapmodify

You can use the `ldapmodify` tool to modify entries from the command line or by using an LDIF file that has the `changetype:modify` directive and value. If you have more than one change, you can separate them using the - (dash) symbol.

To Modify an Attribute from the Command Line

1. Use the `ldapsearch` tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the `ldapmodify` command to change attributes from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be changed using the replace directive. In this example, we change the telephone number of a specific user entry. When you are done typing, you can press **CTRL-D** (Unix EOF escape sequence) twice or **CTRL-Z** (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

To Modify Multiple Attributes in an Entry from the Command Line

1. Use the `ldapsearch` tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the `ldapmodify` command to change attributes from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be changes using the add and replace directive.

   In this example, we add the `postOfficeBox` attribute, change the mobile and telephone numbers of a specific user entry. The `postOfficeBox` attribute must be present in your schema to allow the addition. The three changes are separated by a dash ("-"). When you are done typing, you can press **CTRL-D** (Unix EOF escape sequence) twice or **CTRL-Z** (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postOfficeBox
postOfficeBox: 111
-
replace: mobile
mobile: +1 039 831 3737
-
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

To Add an Attribute from the Command Line

- Use the `ldapmodify` command from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be added using the add directive. In this example, we add another value for the `cn` attribute, which is multi-valued. When you are done typing, you can press **CTRL-D** (UNIX EOF escape sequence) twice to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: cn
cn: Sally Tea Tree
```

An error could occur if the attribute is single-valued, if the value already exists, if the value does not meet the proper syntax, or if the value does not meet the entry's objectclass requirements. Also, make sure there are no trailing spaces after the attribute value.

To Add an Attribute Using the Language Subtype

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

```
$ bin/ldapmodify -h server.example.com -p 389 -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postalAddress; lang-ko
postalAddress; lang-ko:Byung-soon Kim$2020-14 Seoul
```

To Add an Attribute Using the Binary Subtype

The Directory Server provides support for attributes using binary subtypes, which are typically used for certificates or JPEG images that could be stored in an entry. The operation must specifically match the subtype for successful operation. The version directive with a value of "1" must be used for binary subtypes. Typical binary attribute types are `userCertificate` and `jpegPhoto`.

- Use the `ldapmodify` command to add an attribute with a binary subtype. The attribute points to the file path of the certificate.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
version: 1
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate;binary
userCertificate;binary:<file:///path/to/cert
```

To Delete an Attribute

Use `ldapmodify` with the LDIF delete directive to delete an attribute.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: employeeNumber
```

To Delete One Value from an Attribute with Multiple Values

You can use the LDIF delete directive to delete a specific attribute value from an attribute. For this example, assuming you have multiple values of `cn` in an entry (e.g., `cn: Sally Tree`, `cn: Sally Tea Tree`).

- Use `ldapmodify` to delete a specific attribute of a multi-valued pair, then specify the attribute pair that you want to delete. In this example, we keep `cn:Sally Tree` and delete the `cn: Sally Tea Tree`.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: cn
cn: Sally Tea Tree
```

To Rename an Entry

Renaming an entry involves changing the relative distinguished name (RDN) of an entry. You cannot rename a RDN if it has children entries as this violates the LDAP protocol.

- Use the `ldapmodify` tool to rename an entry. The following command changes `uid=user.14` to `uid=user.2014` and uses the changetype, `newrdn`, and `deleteoldrdn` directives.

```
$ bin/ldapmodify
dn: uid=user.14,ou=People,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 1
```

To Move an Entry Within a Directory Server

You can use `ldapmodify` to move an entry from one base DN to another base DN. Before running the `ldapmodify` command, you must assign access control instructions (ACIs) on the parent entries. The source parent entry must have an ACI that allows export operations: `allow(export)`. The target parent entry must have an ACI that allows import operations: `allow(import)`. For more information on access control instructions, see *Working with Access Control*.

- Use the `ldapmodify` command to move an entry from the Contractor branch to the `ou=People` branch.

```
$ bin/ldapmodify
dn: uid=user.14,ou=contractors,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 0
newsuperior: ou=People,dc=example,dc=com
```

To Move an Entry from One Machine to Another

The Directory Server provides a tool, `move-subtree`, to move a subtree or one entry on one machine to another. The subtree or entry must exist on the source server and must not be present on the target server. The source server must also support the 'real attributes only' request control. The target server must support the Ignore NO-USER-MODIFICATION request control.

Note: The `move-subtree` tool moves a subtree or multiple entries from one machine to another. The tool does not copy the entries. Once the entries are moved, they are no longer present on the source server.

- Use the `move-subtree` tool to move an entry (e.g., `uid=test.user,ou=People,dc=example,dc=com`) from the source host to the target host.

```
$ bin/move-subtree --sourceHost source.example.com --sourcePort 389 \
 --sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password
 \
 --targetHost target.example.com --targetPort 389 \
 --targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password
 \
 --entryDN uid=test.user,ou=People,dc=example,dc=com
```

To Move Multiple Entries from One Machine to Another

The `move-subtree` tool provides the ability to move multiple entries listed in a DN file from one machine to another. Empty lines and lines beginning with the octothorpe character (#) will be ignored. Entry DNs may optionally be prefixed with `dn:` , but long DNs cannot be wrapped across multiple lines.

1. Open a text file, enter a list of DNs, one DN per line, and then save the file. You can also use the `ldapsearch` command with the special character "1.1" to create a file containing a list of DNs that you want to move. The following example searches for all entries that match (`department=Engineering`) and returns only the DNs that match the criteria. The results are re-directed to an output file, `test-dns.ldif`:

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com \
     --searchScope sub "(department=Engineering)" "1.1" > test-dns.ldif
   ```
2. Run the `move-subtree` tool with the --entryDNFile option to specify the file of DNs that will be moved from one machine to another.

   ```
   $ bin/move-subtree --sourceHost source.example.com --sourcePort 389 \
     --sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password
    \
     --targetHost target.example.com --targetPort 389 \
     --targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password
    \
     --entryDNFile /path/to/file/test-dns.ldif
   ```
3. If an error occurs with one of the DNs in the file, the output message shows the error. The `move-subtree` tool will continuing processing the remaining DNs in the file.

   ```
   An error occurred while communicating with the target server: The entry
   uid=user.2,ou=People,dc=example,dc=com cannot be added because an entry with
    that name
   already exists
   Entry uid=user.3,ou=People,dc=example,dc=com was successfully moved from
   source.example.com:389 to target.example.com:389
   Entry uid=user.4,ou=People,dc=example,dc=com was successfully moved from
   source.example.com:389 to target.example.com:389
   ```

Working with the Parallel-Update Tool

The PingDirectory Server provides a `parallel-update` tool, which reads change information (add, delete, modify, and modify DN) from an LDIF file and applies the changes in parallel. This tool is a multi-threaded version of the `ldapmodify` tool that is designed to process a large number of changes as quickly as possible.

The `parallel-update` tool provides logic to prevent conflicts resulting from concurrent operations targeting the same entry or concurrent operations involving hierarchically-dependent entries (for example, modifying an entry after it has been added, or adding a child after its parent). The tool also has a retry capability that can help ensure that operations are ultimately successful even when interdependent operations are not present in the correct order in the LDIF file (for example, the change to add a parent entry is provided later in the LDIF file than a change to add a child entry).

After the tool has applied the changes and reaches the end of the LDIF file, it automatically displays the update statistics described in the following table

Table 18: Parallel-Update Tool Result Statistics

Processing Statistic	Description
Attempts	Number of update attempts
Successes	Number of successful update attempts
Rejects	Number of rejected updates
ToRetry	Number of updates that will be retried
AvgOps/S	Average operations per second
RctOps/S	Recent operations per second. Total number of operations from the last interval of change updates.
AvgDurMS	Average duration in milliseconds
RctDurMS	Recent duration in milliseconds. Total duration from the last interval of change updates.

To Run the Parallel-Update Tool

1. Create an LDIF file with your changes. The third change will generate a rejected entry because its `userPassword` attribute contains an encoded value, which is not allowed.

```
dn:uid=user.2,ou=People,dc=example,dc=com
changetype: delete

dn:uid=user.99,ou=People,dc=example,dc=com
changetype: moddn
newrdn: uid=user.100
deleteoldrdn: 1

dn:uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==

dn:uid=user.100,ou=People,dc=example,dc=com
changetype: modify
replace: st
st: TX
-
replace: employeeNumber
employeeNumber: 100
```

2. Use `parallel-update` to apply the changes in the LDIF file to a target server. In this example, we use ten concurrent threads. The optimal number of threads depends on your underlying system. The --ldifFile and --rejectFile options are also required.

```
$ bin/parallel-update --hostname 127.0.0.1 \
  --ldifFile changes.ldif --rejectFile reject.ldif --numThreads 10
```

```
Reached the end of the LDIF file
Attempts Successes Rejects ToRetry AvgOps/S RctOps/S AvgDurMS RctDurMS
-------- --------- ------- ------- -------- -------- -------- ---------
```

```
       4          3        1        0         3         3        26        26
All processing complete Attempted 4 operations in 1 seconds
```

3. View the rejects file for any failed updates.

```
# ResultCode=53, Diagnostic Message=Pre-encoded passwords are not allowed
 for
# the password attribute userPassword
dn: uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==
```

Working with the Watch-Entry Tool

The PingDirectory Server provides a `watch-entry` tool, which demonstrates replication or synchronization latency by watching an LDAP entry for changes. If the entry changes, the background of modified attributes will temporarily be red. Attributes can also be directly modified with this tool as well.

To Run the Watch-Entry Tool

- Perform the following to connect to `server.example.com` as `uid=admin,dc=example,dc=com` and watch entry `uid=kate,ou=people,dc=example,dc=com` for changes:

```
$ bin/watch-entry --hostname server.example.com --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --entryDN uid=user,ou=people,dc=example,dc=com
```

Working with LDAP Transactions

The PingDirectory Server provides support for *batched transactions*, which are processed all at once at commit time. Applications developed to perform batched transactions should include as few operations in the transaction as possible. The changes are not actually processed until the commit request is received. Therefore, the client cannot know whether the changes will be successful until commit time. If any of the operations fail, then the entire set of operations fails.

Batched transactions are write operations (add, delete, modify, modify DN, and password modify) that are processed as a single atomic unit when the commit request is received. If an abort request is received or an error occurs during the commit request, the changes are rolled back. The batched transaction mechanism supports the standard LDAP transaction implementation based on RFC 5805. It is not currently possible to process a transaction that requires changes to be processed across multiple servers or multiple Directory Server backends.

Directory servers may limit the set of controls that are available for use in requests that are part of a transaction. RFC 5805 section 4 indicates that the following controls may be used in conjunction with the transaction specification request control: assertion request control, manageDsaIT request control, pre-read request control, and post-read request control. The proxied authorization v1 and v2 controls cannot be included in requests that are part of a transaction, but they can be included in the start transaction request to indicate that all operations within the transaction should be processed with the specified authorization identity.

The PingDirectory Server supports the following additional controls in conjunction with operations included in a transaction: account usable request control, hard delete request control, intermediate client request control, password policy request control, replication repair request control, soft delete request control, soft deleted entry access request control, subtree delete request control, and undelete request control.

To Request a Batched Transaction Using ldapmodify

1. Use the `ldapmodify` tool's --useTransaction option. It provides a mechanism for processing multiple operations as part of a single batched transaction. Create a batch text file with the changes that you want to apply as a single atomic unit:

```
dn:uid=user.3,ou=People,dc=example,dc=com
changetype: delete
dn:uid=user.1,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 383 288 1090
```

2. Use `ldapmodify` with the --useTransaction and --filename options to run the batched transaction.

```
$ bin/ldapmodify --useTransaction --filename test.ldif
```

```
#Successfully created a transaction with transaction ID 400
#Processing DELETE request for uid=user.3,ou=People,dc=example,dc=com
#DELETE operation successful for DN uid=user.3,ou=People,dc=example,dc=com
#This operation will be processed as part of transaction 400
#Processing MODIFY request for uid=user.1,ou=People,dc=example,dc=com
#MODIFY operation successful for DN uid=user.1,ou=People,dc=example,dc=com
#This operation will be processed as part of transaction 400
#Successfully committed transaction 400
```

Chapter

12

Working with Virtual Attributes

Topics:

The PingDirectory Server provides mechanisms to support virtual attributes in an entry. Virtual attributes are abstract, dynamically generated attributes that are invoked through an LDAP operation, such as `ldapsearch`, but are not stored in the Directory Server backend. While most virtual attributes are operational attributes, providing processing-related information that the server requires, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your directory server requirements.

This chapter presents the following topics:

Overview of Virtual Attributes

The PingDirectory Server allows its entries to hold virtual attributes. Virtual attributes are dynamically generated attributes that are invoked through an LDAP operation, such as `ldapsearch`, but are not stored in the Directory Server backend. Most virtual attributes are operational attributes, providing processing-related information that the server requires. However, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your requirements.

Viewing the List of Default Virtual Attributes

The Directory Server has a default set of virtual attributes that can be viewed using the `dsconfig` tool. Some virtual attributes are enabled by default and are useful for most applications. You can easily enable or disable each virtual attribute using the `dsconfig` tool.

The default set of virtual attributes are described in the table below. You can enable or disable these attributes using the `dsconfig` tool.

Table 19: Virtual Attrbutes

Virtual Attributes	Description
ds-entry-checksum	Generates a simple checksum of an entry’s contents, which can be used with an LDAP assertion control to ensure that the entry has not been modified since it was last retrieved.
ds-instance-name	Generates the name of the Directory Server instance from which the associated entry was read. This virtual attribute can be useful in load-balancing environments to determine the instance from which an entry was retrieved.
entryDN	Generates an `entryDN` operational attribute in an entry that holds a normalized copy of the entry's current distinguished name (DN). Clients can use this attribute in search filters.
hasSubordinates	Creates an operational attribute that has a value of TRUE if the entry has subordinate entries.
isMemberOf	Generates an `isMemberOf` operational attribute that contains the DNs of the groups in which the user is a member.
numSubordinates	Generates an operational attribute that returns the number of child entries. While there is no cost if this operational attribute is enabled, there could be a performance cost if it is requested. Note that this operational attribute only returns the number of immediate children of the node.
subschemaSubentry	A special entry that provides information in the form of operational attributes about the schema elements defined in the server. It identifies the location of the schema for that part of the tree. ldapSyntaxes - set of attribute syntaxes matchingRules - set of matching rules matchingRuleUse - set of matching rule uses attributeTypes - set of attribute types objectClasses - set of object classes nameForms - set of name forms dITContentRules - set of DIT content rules dITStructureRules - set of DIT structure rules

Virtual Attributes	Description
User Defined Virtual Attribute	Generates virtual attributes with user-defined values in entries that match the criteria defined in the plug-in's configuration. User-defined virtual attributes are intended to specify a hard-coded value for entries matching a given set of criteria.
Virtual Static Member	Generates a member attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the `Virtual Static Member` attribute as it is an advanced property and modifying it can lead to undesirable side effects.
Virtual Static Uniquemember	Generates a `uniqueMember` attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the `Virtual Static Uniquemember` attribute as it is an advanced property and modifying it can lead to undesirable side effects.

To View the List of Default Virtual Attributes Using dsconfig Non-Interactive Mode

- Use `dsconfig` to view the virtual attributes.

```
$ bin/dsconfig list-virtual-attributes
```

Viewing Virtual Attribute Properties

Each virtual attribute has basic properties that you can view using the `dsconfig` tool. The complete list of properties is described in the *PingDirectory Server Configuration Reference*. Some basic properties are as follows:

- **Description**. A description of the virtual attribute.
- **Enabled**. Specifies whether the virtual attribute is enabled for use.
- **Base-DN**. Specifies the base DNs for the branches containing entries that are eligible to use this virtual attribute. If no values are given, the server generates virtual attributes anywhere in the server.
- **Group-DN**. Specifies the DNs of the groups whose members can use this virtual attribute. If no values are given, the group membership is not taken into account when generating the virtual attribute. If one or more group DNs are specified, then only members of those groups are allowed to have the virtual attribute.
- **Filter**. Specifies the filters that the server applies to entries to determine if they require virtual attributes. If no values are given, then any entry is eligible to have a virtual attribute value generated.

To View Virtual Attribute Properties

- Use `dsconfig` to view the properties of a virtual attribute.

```
$ bin/dsconfig get-virtual-attribute-prop --name isMemberOf
```

Enabling a Virtual Attribute

You can enable a virtual attribute using the `dsconfig` tool. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Enable a Virtual Attribute using dsconfig Interactive Mode

1. Use `dsconfig` to enable a virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your Directory Server: 1 for LDAP, 2 for SSL, 3 for StartTLS.

   ```
    bin/dsconfig
   ```
2. On the Directory Server main menu, type `o` to change the object menu, and then type the number corresponding to **Standard**.
3. On the Directory Server main menu, type the number corresponding to virtual attributes.
4. On the **Virtual Attribute management** menu, type the number to view and edit an existing virtual attribute.
5. From the list of existing virtual attributes on the system, select the virtual attribute to work with. For this example, type the number corresponding to the `numSubordinates` virtual attribute.
6. On the **numSubordinates Virtual Attribute Properties** menu, type the number to enable the virtual attribute. On the **Enabled Property** menu for the `numSubordinates` virtual attribute, type the number to change the value to TRUE.
7. On the **numSubordinates Virtual Attribute Properties** menu, type `f` to apply the changes.
8. Verify that the virtual attribute is enabled. Note that this example assumes you have configured the group entries.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(ou=People)" numSubordinates
   ```

   ```
   dn: ou=People,dc=example,dc=com
   numSubordinates: 1000
   ```

To Enable a Virtual Attribute Using dsconfig Non-Interactive Mode

- Use `dsconfig` to enable the `numSubordinates` virtual attribute.

  ```
  $ bin/dsconfig set-virtual-attribute-prop \
    --name numSubordinates --set enabled:true
  ```

Creating User-Defined Virtual Attributes

User-defined virtual attributes allow you to specify an explicit value to use for the virtual attribute. There are no restrictions on the length of the value for a user-defined virtual attribute. You must only ensure that the new virtual attribute conforms to your schema, otherwise you will see an error message when you configure it.

You can define your virtual attributes using the `dsconfig` tool on the Standard object menu. Only the value property is specific to the user-defined virtual attribute. All the other properties are common across all kinds of virtual attributes, which include the following:

- **enabled** -- Indicates whether the virtual attribute should be used.
- **attribute-type** -- The attribute type for the virtual attribute that will be generated.
- **base-dn**, **group-dn**, **filter** -- May be used to select which entries are eligible to contain the virtual attribute.
- **client-connection-policy** -- May be used to configure who can see the virtual values.
- **conflict-behavior** -- Used to indicate how the server should behave if there are one or more real values for the same attribute type in the same entry. The server can either return only the real value(s), only the virtual value(s), or merge both real and virtual values.
- **require-explicit-request-by-name** -- Used to indicate whether the server should only generate values for the virtual attribute if it was included in the list of requested attributes.
- **multiple-virtual-attribute-evaluation-order-index**, **multiple-virtual-attribute-merge-behavior** -- Used to control the behavior the server should exhibit if multiple virtual attributes may be used to contribute values to the same attribute.

To Create a User-Defined Virtual Attribute in Interactive Mode

The following example shows how to create a user-defined virtual attribute that assigns an Employee Password Policy to any entry that matches the filter "`(employeeType=employee)`".

1. Run `dsconfig` to configure the user-defined virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your Directory Server: 1 for LDAP, 2 for SSL, 3 for StartTLS.
2. On the Directory Server main menu, type `o` to change the object menu, and then type the number to select Standard.
3. On the Directory Server main menu, type the number corresponding to virtual attributes.
4. On the **Virtual Attribute management** menu, type the number to create a new virtual attribute.
5. Next, you can use an existing virtual attribute as a template for your new attribute, or your can create a new attribute from scratch. In this example, type `n` to create a new Virtual Attribute from scratch.
6. On the **Virtual Attribute Type** menu, enter a number corresponding to the type of virtual attribute that you want to create. In this example, type the number corresponding to User Defined Virtual Attribute.
7. Next, enter a name for the new virtual attribute. In this example, enter "`Employee Password Policy Assignment`."
8. On the **Enabled Property** menu, enter the number to set the property to true (enable).
9. On the **Attribute-Type Property** menu, type the `attribute-type` property for the new virtual attribute. You can enter the OID number or attribute name. The `attribute-type` property must conform to your schema. For this example, type "`ds-pwp-password-policy-dn`".
10. Enter the value for the virtual attribute, and then press Enter or Return to continue. In this example, enter `cn=Employee Password Policy,cn=Password Policies,cn=config`, and then type Enter or Return to continue.
11. On the **User Defined Virtual Attributes** menu, enter a description for the virtual attribute. Though optional, this step is useful if you plan to create a lot of virtual attributes. Enter the option to change the value, and then type a description of the virtual attribute. In this example, enter: `Virtual attribute that assigns the Employee Password Policy to all entries that match (employeeType=employee).`
12. On the **User Defined Virtual Attribute** menu, type the number corresponding to the filter.
13. On the **Filter Property** menu, enter the option to add one or more filter properties, type the filter, and then press **Enter** to continue. In this example, type `(employeeType=employee)`. Press the number to use the filter value entered.
14. On the **User Defined Virtual Attribute** menu, type `f` to finish creating the virtual attribute.
15. Verify that the attribute was created successfully. Add the `employeeType=employee` attribute to an entry (e.g., `uid=user.0`) using `ldapmodify`. Add the `employeeType=contractor` attribute to another entry (e.g., `uid=user.1`).
16. Use `ldapsearch` to search for the user with the `employeeType=employee` attribute (e.g., `uid=user.0`). You will notice the `ds-pwp-password-policy-dn` attribute has the assigned password policy as its value.

    ```
    $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
      ds-password-policy-dn
    ```

    ```
    dn: uid=user.0,ou=People,dc=example,dc=com
    ds-pwp-password-policy-dn: cn=Employee Password Policy,cn=Password
     Policies,cn=config
    ```

17. Run `ldapsearch` again using the filter "`(uid=user.1)`", the `ds-pwp-password-policy-dn` attribute will not be present in the entry, because the entry has the attribute, `employeeType=contractor`.

    ```
    $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.1)" \
      ds-password-policy-dn
    ```

    ```
    dn: uid=user.1,ou=People,dc=example,dc=com
    ```

To Create a User-Defined Virtual Attribute Using dsconfig in Non-Interactive Mode

- You can also use `dsconfig` in non-interactive command-line mode to create a virtual attribute. The following command sets up the Employee Password Policy Assignment virtual attribute introduced in the previous section:

```
$ bin/dsconfig create-virtual-attribute \
  --name "Employee Password Policy Assignment" \
  --type user-defined \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "filter:(employeeType=employee)" \
  --set "value:cn=Employee Password Policy,cn=Password Policies,cn=config"
```

Creating Mirror Virtual Attributes

The PingDirectory Server provides a feature to mirror the value of another attribute in the same entry or mirror the value of the same or a different attribute in an entry referenced by the original entry. For example, given a DIT where users have a manager attributed with a value of the DN of the employee as follows:

```
dn: uid=apeters,ou=people,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
uid: apeters
... (more attributes) ...
```

You can set up a mirror virtual attribute, so that the returned value for the `managerName` virtual attribute can be the `cn` value of the entry referenced by the `manager` attribute as follows:

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=apeters)" \
dn: uid=apeters,ou=people,dc=example,dc=com

objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
managerName: John Doe
uid: apeters
... (more attributes not shown) ...
```

To Create a Mirror Virtual Attribute in Non-Interactive Mode

You can also use `dsconfig` in non-interactive command-line mode to create a mirror virtual attribute. The following example sets up the `managerName` virtual attribute introduced in the previous section:

1. Update the schema to define the `managerName` attribute. In a text editor, create a file with the following schema definition for the attribute and save it as `98-myschema.ldif`, for example, in the `<server-root>/config/schem`a folder. You can optionally add the attribute to an object class.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubent
ry
objectClass: subschema attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.4 NAME
 ‘managerName’
  EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX
 1.3.6.1.4.1.1466.115.121.1.44{256}
  X-ORIGIN ‘Directory Server Example’ )
```

2. Restart the Directory Server.

   ```
   $ bin/stop-server --restart
   ```

3. Use `dsconfig` to create the virtual attribute.

   ```
   $ bin/dsconfig create-virtual-attribute \
     --name "managerName" \
     --type mirror \
     --set "description:managerName from manager cn" \
     --set enabled:true \
     --set attribute-type:managerName \
     --set source-attribute:cn \
     --set source-entry-dn-attribute:manager
   ```

4. Verify the mirror virtual attribute by searching for an entry.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=apeters)"
   ```

   ```
   dn: uid=apeters,ou=People,dc=example,dc=com
   ... (attributes) ...
   manager: uid=jdoe,ou=People,dc=example,dc=com
   managerName: John Doe
   ```

Editing a Virtual Attribute

You can edit virtual attributes using the `dsconfig` tool. You must ensure that the virtual attribute conforms to your plug-in schema, otherwise you will see an error message when you edit the virtual attribute. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Edit a Virtual Attribute Using dsconfig in Non-Interactive Mode

- Use `dsconfig` to change a property's value.

  ```
  $ bin/dsconfig set-virtual-attribute-prop --name dept-number \
    --set "value:111"
  ```

Deleting a Virtual Attribute

You can delete virtual attributes using the `dsconfig` tool. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Delete a Virtual Attribute

- Use `dsconfig` to delete an existing virtual attribute.

  ```
  $ bin/dsconfig delete-virtual-attribute --name dept-number
  ```

Chapter 13

Working with Groups

Topics:

LDAP groups are special types of entries that represent collections of users. Groups are often used by external clients, for example, to control who has access to a particular application or features. They may also be used internally by the server to control its behavior. For example, groups can be used by the access control, criteria, or virtual attribute subsystems.

The specific ways in which clients create and interact with a particular group depends on the type of group being used. In general, there are three primary ways in which clients attempt to use groups:

To determine whether a specified user is a member of a particular group.
To determine the set of groups in which a specified user is a member.
To determine the set of all users that are members of a particular group.

This chapter provides an overview of Directory Server groups concepts and provides procedures on setting up and querying groups in the Directory Server.

Overview of Groups

The Directory Server provides the following types of groups:

- **Static Groups**. A static group is an entry that contains an explicit list of member or uniquemember attributes, depending on its particular structural object class. Static groups are ideal for relatively small, infrequently changing elements. Once the membership list grows, static groups become more difficult to manage as any change in a member base DN must also be changed in the group. Static groups use one of three structural object classes: `groupOfNames`, `groupOfUniqueNames`, and `groupOfEntries`.

  The Directory Server also supports nested groups, in which a parent group entry contains child attributes whose DNs reference another group. Nested groups are a flexible means to organize entries that provide inherited group membership and privileges. To maintain good performance throughput, a group cache is enabled by default. The cache supports static group nesting that includes other static, virtual static, and dynamic groups.
- **Dynamic Groups**. A dynamic group has its membership list determined by search criteria using a LDAP URL. Dynamic groups solve the scalability issues encountered for static groups as searches are efficient, constant-time operations. However, if searches range over a very large set of data, performance could be affected.
- **Virtual Static Groups**. A virtual static group is a combination of both static and dynamic groups, in which each member in a group is a virtual attribute that is dynamically generated when invoked. Virtual static groups solve the scalability issues for clients that can only support static groups and are best used when the application targets a search operation for a specific member. Virtual static groups are not good for applications that need to retrieve the entire membership list as the process for constructing the entire membership list can be expensive.

About the isMemberOf and isDirectMemberOf Virtual Attribute

The existence of both static, nested, dynamic, and virtual static groups can make it unnecessarily complex to work with groups in the server, particularly because the ways you interact with them are so different. And the fact that static groups can use three different structural object classes (not counting the auxiliary class for virtual static groups) does not make things any easier.

To make group operations simpler, the PingDirectory Server provides the ability to generate either an `isMemberOf` and `isDirectMemberOf` virtual attributes in user entries. These attributes dramatically simplify the process for making group-related determinations in a manner that is consistent across all types of groups.

The value of the `isMemberOf` virtual attribute is a list of DNs of all groups (including static, nested, dynamic, and virtual static groups) in which the associated user is a member. The value of the `isDirectMemberOf` virtual attribute is a subset of the values of `isMemberOf`, which represents the groups for which the entry is an explicit or direct member. Both are enabled by default.

Because the `isMemberOf` and `isDirectMemberOf` are operational attributes, only users who specifically have been granted the privilege can see it. The default set of access control rules do not allow any level of access to user data. The only access that is granted is what is included in user-defined access control rules, which is generally given to a `uid=admin` administrator account. It is always a best practice to restrict access to operational and non-operational attributes to the minimal set of users that need to see them. The root bind DN, `cn=Directory Manager`, has the privilege to view operational attributes by default.

To determine whether a user is a member of a specified group using the `isMemberOf` virtual attribute, simply perform a base-level search against the user's entry with an equality filter targeting the `isMemberOf` attribute with a value that is the DN of the target group. The following table illustrates this simple base-level search:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)
Requested Attributes	1.1

If this search returns an entry, then the user is a member of the specified group. If no entry is returned, then the user is not a member of the given group.

To determine the set of all groups in which a user is a member, simply retrieve the user's entry with a base-level search and include the `isMemberOf` attribute:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(objectclass=*)
Requested Attributes	isMemberOf

To determine the set of all members for a specified group, issue a subtree search with an equality filter targeting the `isMemberOf` attribute with a value that is the DN of the target group and requesting the attributes you wish to have for member entries:

Base DN	ou=People,dc=example,dc=com
Scope	sub
Filter	(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)
Requested Attributes	cn, mail

The `isDirectMemberOf` virtual attribute can be used in the examples above in place of `isMemberOf` if you only need to find groups that users are an actual member of. You must use `isMemberOf` for nested group membership.

Note that if this filter targets a dynamic group using an unindexed search, then this may be an expensive operation. However, it will not be any more expensive than retrieving the target group and then issuing a search based on information contained in the member URL.

For static groups, this approach has the added benefit of using a single search to retrieve information from all user entries, whereas it would otherwise be required to retrieve the static group and then perform a separate search for each member's entry.

Using Static Groups

A static group contains an explicit membership list where each member is represented as a DN-valued attribute. There are three types of static groups supported for use in the Directory Server:

- **groupOfNames**. A static group that is defined with the `groupOfNames` structural object class and uses the `member` attribute to hold the DNs of its members. RFC 4519 requires that the `member` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `member` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
cn: Test Group
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```

- **groupOfUniqueNames**. A static group that is defined with the `groupOfUniqueNames` structural object class and uses the `uniquemember` attribute to hold the DNs of its members. RFC 4519 requires that the `uniquemember` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `uniquemember` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfUniqueNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: Test Group
uniquemember: uid=user.1,ou=People,dc=example,dc=com
uniquemember: uid=user.2,ou=People,dc=example,dc=com
uniquemember: uid=user.3,ou=People,dc=example,dc=com
```

- **groupOfEntries**. A static group that is defined with the `groupOfEntries` object class and uses the `member` attribute to hold the DNs of its members. This group specifies that the `member` attribute is optional to ensure that the last member can be removed from the group. Although the draft proposal (draft-findlay-ldap-groupofentries-00.txt) has expired, the Directory Server supports this implementation. The following entry depicts a group defined with the `groupOfEntries` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfEntries
cn: Test Group
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```

Creating Static Groups

You can configure a static group by adding it using an LDIF file. Static groups contain a membership list of explicit DNs specified by the `uniquemember` attribute.

To Create a Static Group

1. Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniquenames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your server, then you can add it in the same file. When done, save the file as `static-group.ldif`. The following example LDIF file creates two groups, `cn=Development` and `cn=QA`.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Development,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Development
ou: groups
uniquemember: uid=user.14,ou=People,dc=example,dc=com
uniquemember: uid=user.91,ou=People,dc=example,dc=com
uniquemember: uid=user.180,ou=People,dc=example,dc=com

dn: cn=QA,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: QA
ou: groups
uniquemember: uid=user.0,ou=People,dc=example,dc=com
uniquemember: uid=user.1,ou=People,dc=example,dc=com
uniquemember: uid=user.2,ou=People,dc=example,dc=com
```

2. Use `ldapmodify` to add the group entries to the server.

```
$ bin/ldapmodify --defaultAdd --filename static-group.ldif
```

3. Verify the configuration by using the virtual attribute `isDirectMemberOf` that checks membership for a non-nested group. By default, the virtual attribute is disabled by default, but you can enable it using `dsconfig`.

```
$ bin/dsconfig set-virtual-attribute-prop --name isDirectMemberOf --set
 enabled:true
```

4. Use `ldapsearch` to specifically search the `isDirectMemberOf` virtual attribute to determine if `uid=user.14` is a member of the `cn=Development` group. In this example, assume that administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isDirectMemberOf
```

```
dn: uid=user.14,ou=People,dc=example,dc=com
isDirectMemberOf: cn=Development,ou=groups,dc=example,dc=com
```

5. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an `aci` attribute in an LDIF file, and save the file as `dev-group-aci.ldif`. Add the file using the `ldapmodify` tool. You can create a similar ACI for the QA group, which is not shown in this example.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr != "cn || sn || uid")
  (targetfilter ="(ou=Development)")
  (version 3.0; acl "Dev Group Permissions";
    allow (write) (groupdn = "ldap:///
cn=Development,ou=groups,dc=example,dc=com");)
```

6. Add the file using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename dev-group-aci.ldif
```

To Add a New Member to a Static Group

- To add a new member to the group, add a new value for the `uniquemember` attribute that specifies the DN of the user to be added. The following example adds a new `uniquemember`, `user.4`:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
add: uniquemember
uniquemember: uid=user.4,ou=People,dc=example,dc=com
```

To Remove a Member from a Static Group

- To remove a member from a static group, remove that user's DN from the `uniquemember` attribute. The following example removes the DN of `user.1`:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
delete: uniquemember
uniquemember: uid=user.1,ou=People,dc=example,dc=com
```

Searching Static Groups

The following sections describe how to compose searches to determine if a user is a member of a static group, to determine all the static groups in which a user is a member, and to determine all the members of a static group.

To Determine if a User is a Static Group Member

To determine whether a user is a member of a specified group, perform a base-level search to retrieve the group entry with an equality filter looking for the membership attribute of a value equal to the DN of the specified user.

For best performance, you will want to include a specific attribute list (just "cn", or "1.1" request that no attributes be returned) so that the entire member list is not returned. For example, to determine whether the user "`uid=john.doe,ou=People,dc=example,dc=com`" is a member of the `groupOfNames` static group "`cn=Test Group,ou=Groups,dc=example,dc=com`", issue a search with the following criteria:

Table 20: Search Criteria for a Single User's Membership in a Static Group

Base DN	cn=Test Group,ou=Groups,dc=example,dc=com
Scope	base
Filter	(member=uid=john.doe,ou=People,dc=example,dc=com)
Requested Attributes	1.1

If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group. If you do not know the membership attribute for the specified group (it could be either a `member` or `uniqueMember` attribute), then you may want to revise the filter so that it allows either one as follows:

```
(|(member=uid=john.doe,ou=People,dc=example,dc=com)
(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com))
```

- Run a base-level search to retrieve the group entry with an equality filter looking for the membership attribute.

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com"
  --searchScope base "(member=uid=john.doe,ou=People,dc=example,dc=com)"
 "1.1"
```

To Determine the Static Groups to Which a User Belongs

To determine the set of all static groups in which a user is specified as a member, perform a subtree search based at the top of the DIT. The search filter must be configured to match any type of static group in which the specified user is a member.

For example, the following criteria may be used to determine the set of all static groups in which the user, `uid=john.doc,ou=People,dc=example,dc=com`, is a member:

Table 21: Search Criteria for Determining All the Static Groups for a User

Base DN	dc=example,dc=com
Scope	sub
Filter	(|(&(objectClass=groupOfNames) (member=uid=john.doe,ou=People,dc=example,dc=com)) (&(objectClass=groupOfUniqueNames)(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com)) (&(objectClass=groupOfEntries) (member=uid=john.doe,ou=People,dc=example,dc=com)))
Requested Attributes	1.1

Every entry returned from the search represents a static group in which the specified user is a member.

- Run a sub-level search to retrieve the static groups to which a user belongs.

```
$ bin/ldapsearch --baseDN "dc=example,dc=com" --searchScope sub \
  "(|(&(objectClass=groupOfNames)
   (member=uid=john.doe,ou=People,dc=example,dc=com)) \
   (&(objectClass=groupOfUniqueNames)\
   (uniqueMember=uid=john.doe,ou=People,dc=example,dc=com)) \
   (&(objectClass=groupOfEntries) \
   (member=uid=john.doe,ou=People,dc=example,dc=com)))" "1.1"
```

Note: A base level search of the user's entry for `isMemberOf` or `isDirectMemberOf` virtual attributes will give the same results. You can also use the virtual attributes with virtual static groups.

To Determine the Members of a Static Group

To determine all of the members for a static group, simply retrieve the group entry including the membership attribute. The returned entry will include the DNs of all users that are members of that group. For example, the following criteria may be used to retrieve the list of all members for the group `cn=Test Group,ou=Groups,dc=example,dc=com`:

Table 22: Search Criteria for All of the Static Group's Members

Base DN	cn=Test Group,ou=Groups,dc=example,dc=com
Scope	base
Filter	(objectClass=*)
Requested Attributes	member uniqueMember

If you want to retrieve additional information about the members, such as attributes from member entries, you must issue a separate search for each member to retrieve the user entry and the desired attributes.

- Run a base-level search to retrieve all of the members in a static group.

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com" \
  --searchScope base "(objectclass=*)" uniqueMember
```

Note: If you want to retrieve attributes from member entries, it is more efficient to search all users whose `isMemberOf` attribute contains the group DN, returning the attributes desired.

Using Dynamic Groups

Dynamic groups contain a set of criteria used to identify members rather than maintaining an explicit list of group members. If a new user entry is created or if an existing entry is modified so that it matches the membership criteria, then the user will be considered a member of the dynamic group. Similarly, if a member's entry is deleted or if it is modified so that it no longer matches the group criteria, then the user will no longer be considered a member of the dynamic group.

In the Directory Server, dynamic groups include the `groupOfURLs` structural object class and use the `memberurl` attribute to provide an LDAP URL that defines the membership criteria. The base, scope, and filter of the LDAP URL will be used in the process of making the determination, and any other elements present in the URL will be ignored. For example, the following entry defines a dynamic group in which all users below `dc=example,dc=com` with an `employeeType` value of contractor will be considered members of the group:

```
dn: cn=Sales Group,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Sales Group
memberURL: ldap:///dc=example,dc=com??sub?(employeeType=contractor)
```

Assuming that less than 80,000 entries have the `employeeType` of contractor, you need to create the following index definition to evaluate the dynamic group:

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name employeeType --set index-entry-limit:80000 \
  --set index-type:equality
```

Creating Dynamic Groups

You can configure a dynamic group in the same manner as static groups using an LDIF file. Dynamic groups contain a membership list of attributes determined by search filter using an LDAP URL. You must use the `groupOfURLs` object class and the `memberURL` attribute.

To Create a Dynamic Group

1. Assume that `uid=user.15` is not part of any group. Use `ldapsearch` to verify that `uid=user.15` is not part of any group. In a later step, we will add the user to the dynamic group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub
 "(uid=user.15)" ou
```

```
dn: uid=user.15,ou=People,dc=example,dc=com
```

2. Assume for this example that `uid=user.0` has an `ou=Engineering` attribute indicating that he or she is a member of the Engineering department.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.0)"
 ou isMemberOf
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ou: Engineering
```

3. Open a text editor, and then create a dynamic group entry in LDIF. The LDIF defines the dynamic group to include all users who have the `ou=Engineering` attribute. When done, save the file as `add-dynamic-group.ldif`.

```
dn: cn=eng-staff,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfURLs
ou: groups
cn: eng-staff
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(ou=Engineering)
```

4. Use `ldapmodify` to add the group entry to the server.

```
$ bin/ldapmodify --defaultAdd --filename add-dynamic-group.ldif
```

5. Use `ldapsearch` to specifically search the `isMemberOf` virtual attribute to determine if `uid=user.0` is a member of the `cn=Engineering` group or any other group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" isMemberOf
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

6. If your data is relatively small (under 1 million entries), you can search for all users in the group that meet the search criteria (`ou=Engineering`). For very large databases, it is not practical to run a database-wide search for all users as there can be a performance hit on the Directory Server. The following command returns the DNs of entries that are part of the `cn=eng-staff` dynamic group and sorts them in ascending order by the `sn` attribute.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --sortOrder sn \
  "(isMemberOf=cn=eng-staff,ou=groups,dc=example,dc=com)" dn
```

7. Add `uid=user.15` to the `eng-staff` group by adding an `ou=Engineering` attribute to the entry. This step highlights an advantage of dynamic groups: you can make a change in an entry without explicitly adding the DN to the group as you would with static groups. The entry will be automatically added to the `eng-staff` dynamic group.

```
$ bin/ldapmodify
dn: uid=user.15,ou=People,dc=example,dc=com
changetype: modify
```

```
add: ou
ou: Engineering
```

8. Use `ldapsearch` to check if the user is part of the `cn=eng-staff` dynamic group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub
 "(uid=user.15)" isMemberOf
```

```
dn: uid=user.15,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

Searching Dynamic Groups

The following sections describe how to compose searches to determine if a user is a member of a dynamic group, to determine all the dynamic groups in which a user is a member, and to determine all the members of a dynamic group.

To Determine if a User is a Dynamic Group Member

To determine whether a user is a member of a specific dynamic group, you must verify that the user's entry is both within the scope of the member URL and that it matches the filter contained in that URL. You can verify that a user's entry is within the scope of the URL using simple client-side only processing. Evaluating the filter against the entry on the client side can be more complicated. While possible, particularly in clients that are able to perform schema-aware evaluation, a simple alternative is to perform a base-level search to retrieve the user's entry with the filter contained in the member URL.

For example, to determine whether the user `uid=john.doe,ou=People,dc=example,dc=com` is a member of the dynamic group with the above member URL, issue a search with the following criteria:

Table 23: Search Criteria for a Single User's Membership in a Dynamic Group

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(ou=Engineering)
Requested Attributes	1.1

Note that the search requires the user DN to be under the search base defined in the `memberurl` attribute for the user to be a member. If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group.

To Determine the Dynamic Groups to Which a User Belongs

To determine the set of all dynamic groups in which a user is a member, first perform a search to find all dynamic group entries defined in the server. You can do this using a subtree search with a filter of `"(objectClass=groupOfURLs)"`.

You should retrieve the `memberURL` attribute so that you can use the logic described in the previous section to determine whether the specified user is a member of each of those groups. For example, to find the set of all dynamic groups defined in the `dc=example,dc=com` tree, issue a search with the following criteria:

Table 24: Search Criteria for Determining All of the Dynamic Groups for a User

Base DN	dc=example,dc=com
Scope	sub
Filter	(objectClass=groupOfURLs)
Requested Attributes	memberURL

Each entry returned will be a dynamic group definition. You can use the base, scope, and filter of its `memberURL` attribute to determine whether the user is a member of that dynamic group.

To Determine the Members of a Dynamic Group

To determine all members of a dynamic group, issue a search using the base, scope, and filter of the member URL. The set of requested attributes should reflect the attributes desired from the member user entries, or "1.1" if no attributes are needed.

For example, to retrieve the `cn` and `mail` attributes from the group described above, use the following search:

Table 25: Search Criteria for Determining the Members of a Dynamic Group

Base DN	dc=example,dc=com
Scope	sub
Filter	(employeeType=contractor)
Requested Attributes	cn, mail

⚠ **Caution:** Note that this search may be expensive if the associated filter is not indexed or if the group contains a large number of members.

Using Dynamic Groups for Internal Operations

You can use dynamic groups for internal operations, such as ACI or component evaluation. The Directory Server performs the `memberurl` parsing and internal LDAP search; however, the internal search operation may not be performed with access control rules applied to it.

For example, the following dynamic group represents an organization’s employees within the same department:

```
dn: cn=department 202,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: department 202
owner: uid=user.1,ou=people,dc=example,dc=com
owner: uid=user.2,ou=people,dc=example,dc=com
memberURL: ldap:///ou=People,dc=example,dc=com??sub?
  (&(employeeType=employee)(departmentNumber=202))
description: Group of employees in department 202
```

The above group could be referenced from within the ACI at the `dc=example,dc=com` entry. For example:

```
dn:dc=example,dc=com
aci: (targetattr="employeeType")
  (version 3.0; acl "Grant write access to employeeType" ;
    allow (all) groupdn="ldap:///cn=department
 202,ou=groups,dc=example,dc=com";)
```

Any user matching the filter can bind to the server with their entry and modify the `employeeType` attribute within any entry under `dc=example,dc=com`.

Using Virtual Static Groups

Static groups can be easier to interact with than dynamic groups, but large static groups can be expensive to manage and require a large amount of memory to hold in the internal group cache. The Directory Server provides a third type of group that makes it possible to get the efficiency and ease of management of a dynamic group while allowing clients to interact with it as a static group. A *virtual static group* is a type of group that references another group and provides access to the members of that group as if it was a static group.

To create a virtual static group, create an entry that has a structural object class of either `groupOfNames` or `groupOfUniqueNames` and an auxiliary class of `ds-virtual-static-group`. It should also include a `ds-target-group-dn` attribute, whose value is the group from which the virtual static group should obtain its members. For example, the following will create a virtual static group that exposes the members of the `cn=Sales Group,ou=Groups,dc=example,dc=com` dynamic group as if it were a static group:

```
dn: cn=Virtual Static Sales Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
objectClass: ds-virtual-static-group
cn: Virtual Static Sales Group
ds-target-group-dn: cn=Sales Group,ou=Groups,dc=example,dc=com
```

Note that you must also enable a virtual attribute that allows the `member` attribute to be generated based on membership for the target group. A configuration object for this virtual attribute does exist in the server configuration, but is disabled by default. To enable it, issue the following change:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set enabled:true
```

If you want to use virtual static groups with the `groupOfUniqueNames` object class, then you will also need to enable the `Virtual Static uniqueMember` virtual attribute in the same way.

Creating Virtual Static Groups

If your application only supports static groups but has scalability issues, then using a virtual static group could be a possible solution. A virtual static group uses a virtual attribute that is dynamically generated when called after which the operations that determine group membership are passed to another group, such as a dynamic group. You must use the `ds-virtual-static-group` object class and the `ds-target-group-dn` virtual attribute.

Virtual static groups are best used when determining if a single user is a member of a group. It is not a good solution if an application accesses the full list of group members due to the performance expense at constructing the list. If you have a small database and an application that requires that the full membership list be returned, you must also enable the `allow-retrieving-membership` property for the Virtual Static `uniqueMember` virtual attribute using the `dsconfig` tool.

To Create a Virtual Static Group

1. Open a text editor, and then create a group entry in LDIF. The entry contains the `groupOfUniqueNames` object class, but in place of the `uniquemember` attribute is the `ds-target-group-dn` virtual attribute, which is part of the `ds-virtual-static-group` auxiliary object class. When done, save the file as `add-virtual-static-group.ldif`.

   ```
   dn: cn=virtualstatic,ou=groups,dc=example,dc=com
   objectclass: top
   objectclass: groupOfUniqueNames
   objectclass: ds-virtual-static-group
   ou: groups
   cn: virtual static
   ds-target-group-dn: cn=eng-staff,ou=groups,dc=example,dc=com
   ```
2. Use `ldapmodify` to add the virtual static group entry to the server.

   ```
   $ bin/ldapmodify -h server1.example.com -p 389 -D
    "uid=admin,dc=example,dc=com" \
     -w password -a -f add-virtual-static-group.ldif
   ```
3. Use `dsconfig` to enable the Virtual Static `uniqueMember` attribute, which is disabled by default.

   ```
   $ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
    uniqueMember" \
     --set enabled:true
   ```

4. In the previous section, we set up uid=user.0 to be part of the cn=eng-staff dynamic group. Use ldapsearch with the isMemberOf virtual attribute to determine if uid=user.0 is part of the virtual static group.

   ```
   $ bin/ldapsearch -h server1.example.com -p 389 -D "cn=Directory Manager" \
     -w secret -b dc=example,dc=com" "(uid=user.0)" isMemberOf
   ```

   ```
   dn: uid=user.0,ou=People,dc=example,dc=com
   isMemberOf: cn=virtualstatic,ou=groups,dc=example,dc=com
   isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
   ```
5. Use ldapsearch to determine if uid=user.0 is a member of the virtual static group. You should see the returned cn=virtualstatic entry if successful.

   ```
   $ ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w password \
     -b "cn=virtualStatic,ou=Groups,dc=example,dc=com" \
     "(&(objectclass=groupOfUniqueNames) \
     (uniquemember=uid=user.0,ou=People,dc=example,dc=com))"
   ```
6. Next, try searching for a user that is not part of the cn=eng-staff dynamic group (e.g., uid=user.20), nothing will be returned.

   ```
   $ ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w password \
     -b "cn=virtualStatic,ou=Groups,dc=example,dc=com" \
     "(&(objectclass=groupOfUniqueNames) \
     (uniquemember=uid=user.20,ou=People,dc=example,dc=com))"
   ```

Searching Virtual Static Groups

Because virtual static groups behave like static groups, the process for determining whether a user is a member of a virtual static group is identical to that of a member in a static group. Similarly, the process for determining all virtual static groups in which a user is a member is basically the same as the process as that of real static groups in which a user is a member. In fact, the query provided in the static groups discussion returns virtual static groups in addition to real static groups, because the structural object class of a virtual static group is the same as the structural object class for a static group.

You can also retrieve a list of all members of a virtual static group in the same way as a real static group: simply retrieve the member or uniqueMember attribute of the desired group. However, because virtual static groups are backed by dynamic groups and the process for retrieving member information for dynamic groups can be expensive, virtual static groups do not allow retrieving the full set of members by default. The virtual attribute used to expose membership can be updated to allow this with a configuration change such as the following:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set allow-retrieving-membership:true
```

Because this can be an expensive operation, we recommend that the option to allow retrieving virtual static group membership be left disabled unless it is required.

Creating Nested Groups

The PingDirectory Server supports nested groups, where the DN of an entry that defines a group is included as a member in the parent entry. For example, the following example shows a nested static group (e.g., cn=Engineering Group) that has uniquemember attributes consisting of other groups, such as cn=Developers Group and the cn=QA Group respectively.

```
dn: cn=Engineering Group,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers,ou=Groups,dc=example,dc=com
```

```
uniquemember: cn=QA,ou=Groups,dc=example,dc=com
```

Nested group support is enabled by default on the Directory Server. To support nested groups without the performance hit, the Directory Server uses a group cache, which is also enabled by default. The cache supports static group nesting that includes other static, virtual static, and dynamic groups. The Directory Server provides a new monitoring entry for the group cache, `cn=Group Cache,cn=Monitor`.

In practice, nested groups are not commonly used for several reasons. LDAP specifications do not directly address the concept of nested groups, and some servers do not provide any level of support for them. Supporting nested groups in LDAP clients is not trivial, and many directory server-enabled applications that can interact with groups do not provide any support for nesting. If nesting support is not needed in your environment, or if nesting support is only required for clients but is not needed for server-side evaluation (such as for groups used in access control rules, criteria, virtual attributes, or other ways that the server may need to make a membership determination), then this support should be disabled.

To Create Nested Static Groups

1. The following example shows how to set up a nested static group, which is a static group that contains `uniquemember` attributes whose values contain other groups (static, virtual static, or dynamic). Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniquenames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your server, then you can add it in the same file. When done, save the file as `nested-group.ldif`. Assume that the static groups, `cn=Developers Group` and `cn=QA Group`, have been configured.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Engineering Group,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers,ou=groups,dc=example,dc=com
uniquemember: cn=QA,ou=groups,dc=example,dc=com
```

2. Use `ldapmodify` to add the group entry.

```
$ bin/ldapmodify --defaultAdd --filename nested-static-group.ldif
```

3. Verify the configuration by using the `isMemberOf` virtual attribute that checks the group membership for an entry. By default, the virtual attribute is enabled. Use `ldapsearch` to specifically search the `isMemberOf` virtual attribute to determine if `uid=user.14` is a member of the `cn=Development` group. In this example, assume that the administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isMemberOf

dn: uid=user.14,ou=People,dc=example,dc=com
isMemberOf: cn=Development,ou=groups,dc=example,dc=com
```

4. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an ACI in LDIF, and save the file as `eng-group-aci.ldif`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr != "cn || sn || uid")
  (targetfilter ="(ou=Engineering Group)")
  (version 3.0; acl "Engineering Group Permissions";
    allow (write) (groupdn = "ldap:///cn=Engineering
 Group,ou=groups,dc=example,dc=com");)
```

5. Add the file using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename eng-group-aci.ldif
```

Note: When nesting dynamic groups, you cannot include other groups as members of a dynamic group. You can only support "nesting" by including the members of another group with a filter in the member URL. For example, if you have two groups `cn=dynamic1` and `cn=dynamic2`, you can nest one group in another by specifying it in the member URL as follows:

```
cn=dynamic1,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
memberURL: ldap:///dc=example,dc=com??sub?
(isMemberOf=cn=dynamic2,ou=groups,dc=example,dc=com)
```

The members included from the other group using this method are not considered "nested" members and will be returned even when using `isDirectMemberOf` when retrieving the members.

Maintaining Referential Integrity with Static Groups

The Directory Server can automatically update references to an entry whenever that entry is removed or renamed in a process called *referential integrity*. For example, if a user entry is deleted, then referential integrity plugin will remove that user from any static groups in which the user was a member (this is not necessary for dynamic groups, since no explicit membership is maintained). Similarly, if a modify DN operation is performed to move or rename a user entry, then referential integrity updates static groups in which that user is a member with the new user DN.

Referential integrity support is disabled by default, but may be enabled using the `dsconfig` tool as follows:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

Other configuration attributes of note for this plugin include:

- **attribute-type**. This attribute specifies the names or OIDs of the attribute types for which referential integrity will be maintained. By default, referential integrity is maintained for the `member` and `uniqueMember` attributes. Any attribute types specified must have a syntax of either distinguished name (OID "1.3.6.1.4.1.1466.115.121.1.12") or name and optional UID (OID "1.3.6.1.4.1.1466.115.121.1.34"). The specified attribute types must also be indexed for equality in all backends for which referential integrity is to be maintained.
- **base-dn**. This attribute specifies the subtrees for which referential integrity will be maintained. If one or more values are provided, then referential integrity processing will only be performed for entries which exist within those portions of the DIT. If no values are provided (which is the default behavior), then entries within all public naming contexts will be included.
- **log-file**. This attribute specifies the path to a log file that may be used to hold information about the DNs of deleted or renamed entries. If the plugin is configured with a nonzero update interval, this log file helps ensure that appropriate referential integrity processing occurs even if the server is restarted.
- **update-interval**. This attribute specifies the maximum length of time that a background thread may sleep between checks of the referential integrity log file to determine whether any referential integrity processing is required. By default, this attribute has a value of "0 seconds", which indicates that all referential integrity processing is to be performed synchronously before a response is returned to the client. A duration greater than 0 seconds indicates that referential integrity processing will be performed in the background and will not delay the response to the client.

In the default configuration, where referential integrity processing is performed synchronously, the throughput and response time of delete and modify DN operations may be adversely impacted because the necessary cleanup work must be completed before the response to the original operation can be returned. Changing the configuration to use a non-zero update interval alleviates this performance impact because referential integrity processing uses a separate background thread and does not significantly delay the response to delete or modify DN operations.

However, performing referential integrity processing in a background thread may introduce a race condition that may adversely impact clients that delete a user and then immediately attempt to re-add it and establish new group memberships. If referential integrity processing has not yet been completed for the delete, then newly-established group memberships may be removed along with those that already existed for the previous user. Similarly, if the newly-created user is to be a member of one or more of the same groups as the previous user, then attempts by the client to re-establish those memberships may fail if referential integrity processing has not yet removed the previous membership. For this reason, we recommend that the default synchronous behavior be maintained unless the performance impact associated with it is unacceptable and clients are not expected to operate in a manner that may be adversely impacted by delayed referential integrity processing.

Note: The internal operations of the referential integrity plug-in are not replicated. So, in a replicated topology, you must enable the referential integrity plug-in consistently on all servers in the topology to ensure that changes made by the referential integrity plug-in are passed along to a replication server.

For more information about administering the referential integrity plug-in, see Chapter 6, “Configuring the Directory Server” in the *Ping Identity PingDirectory Server Administration Guide*.

Monitoring the Group Membership Cache

The Directory Server logs information at startup about the memory consumed by the group membership cache. This hard-coded cache contains information about all of the group memberships for internal processing, such as ACIs. The group membership cache is enabled by default.

The information about this cache is logged to the standard output log (`server.out`) and the standard error log. When using groups, you can use the log information to tune the server for best performance. For example, at startup the server logs a message like the following to the `server.out` log:

```
[16/Aug/2011:17:14:39.462 -0500] category=JEB severity=NOTICE msgID=1887895587
msg="The database cache now holds 3419MB of data and is 32 percent full"
```

The error log will contain something like the following:

```
[16/Aug/2011:18:40:39.555 -0500] category=EXTENSIONS severity=NOTICE
 msgID=1880555575
msg="'Group cache (174789 static group(s) with 7480151 total memberships and
 1000002
unique members, 0 virtual static group(s), 1 dynamic group(s))' currently
 consumes
149433592 bytes and can grow to a maximum of 149433592 bytes"
```

Using the Entry Cache to Improve the Performance of Large Static Groups

The PingDirectory Server provides an entry cache implementation, which allows for fine-grained control over the kinds of entries that may be held in the cache. You can define filters to specify the entries included in or excluded from the cache, and you can restrict the cache so that it holds only entries with at least a specified number of values for a given set of attributes.

Under most circumstances, we recommend that the Directory Server be used *without* an entry cache. The Directory Server is designed to efficiently retrieve and decode entries from the database in most cases. The database cache is much more space-efficient than the entry cache, and heavy churn in the entry cache can adversely impact garbage collection behavior.

However, if the Directory Server contains very large static groups, such as those containing thousands or millions of members, and clients need to frequently retrieve or otherwise interact with these groups, then you may want to enable an entry cache that holds only large static groups.

In servers containing large static groups, you can define an entry cache to hold only those large static groups. This entry cache should have an include filter that matches only group entries (for example, "`(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfEntries))`"). The filter contains a minimum value count so that only groups with a large number of members (such as those with at least 100 `member` or `uniqueMember` values) will be included. The Directory Server provides an entry cache implementation with these settings although it is disabled by default.

To Enable the Entry Cache

- Run `dsconfig` to enable the entry cache.

```
$ bin/dsconfig set-entry-cache-prop --cache-name "Static Group Entry Cache"
 \
  --set enabled:true
```

To Create Your Own Entry Cache for Large Groups

- You can create your own entry cache for large groups using the `dsconfig create-entry-cache` subcommand.

```
# bin/dsconfig create-entry-cache --type fifo \
  --set enabled:true \
  --set cache-level:10 \
  --set max-entries:175000 \
  --set "include-filter:(objectClass=groupOfUniqueNames)" \
  --set min-cache-entry-value-count:10000 \
  --set min-cache-entry-attribute:uniquemember
```

Monitoring the Entry Cache

You can monitor the memory consumed by your entry cache using the `entry-cache-info` property in the periodic stats logger. You can retrieve the monitor entry over LDAP by issuing a search on `baseDN="cn=monitor"` using `filter="(objectClass=ds-fifo-entry-cache-monitor-entry)"`. For example, the entry might appear as follows:

```
dn: cn=Static Group Entry Cache Monitor,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-fifo-entry-cache-monitor-entry
objectClass: extensibleObject
cn: Static Group Entry Cache Monitor
cacheName: Static Group Entry Cache
entryCacheHits: 6416407
entryCacheTries: 43069073
entryCacheHitRatio: 14
maxEntryCacheSize: 12723879900
currentEntryCacheCount: 1
maxEntryCacheCount: 175000
entriesAddedOrUpdated: 1
evictionsDueToMaxMemory: 0
evictionsDueToMaxEntries: 0
entriesNotAddedAlreadyPresent: 0
entriesNotAddedDueToMaxMemory: 0
entriesNotAddedDueToFilter: 36652665
entriesNotAddedDueToEntrySmallness: 0
lowMemoryOccurrences: 0
percentFullMaxEntries: 0
jvmMemoryMaxPercentThreshold: 75
jvmMemoryCurrentPercentFull: 24
jvmMemoryBelowMaxMemoryPercent: 51
isFull: false
capacityDetails: NOT FULL: The JVM is using 24% of its available memory.
 Entries can be
```

```
added to the cache until the overall JVM memory usage reaches the configured
 limit of
75%. Cache has 174999 remaining entries before reaching the configured limit
 of 175000.
```

By default, the entry cache memory is set to 75%, with a maximum of 90%.

Tuning the Index Entry Limit for Large Groups

The Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the DIT. You can specify an index entry limit property, which defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (which is 4000 by default), then you must rebuild the indexes using the `rebuild-index` tool as follows:

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index objectclass
```

In the majority of Directory Server environments, the default index entry limit value of 4000 entries should be sufficient. However, group-related processing, it may be necessary to increase the index entry limit. For directories containing more than 4000 groups with the same structural object class (i.e., more than 4000 entries, 4000 `groupOfUniqueNames` entries, 4000 `groupOfEntries` entries, or 4000 `groupOfURLs` entries), then you may want to increase the index entry limit for the `objectClass` attribute so that it has a value larger than the maximum number of group entries of each type. Set `index-entry-limit` property using a command line like the following:

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
  --index-name objectClass --set index-entry-limit:175000
```

As an alternative, a separate backend may be created to hold these group entries, so that an unindexed search in that backend yields primarily group entries. If you make no changes, then the internal search performed at startup to identify all groups and any user searches looking for groups of a given type may be very expensive.

For directories in which any single user may be a member of more than 4000 static groups of the same type, you may need to increase the index entry limit for the `member` and/or `uniqueMember` attribute to a value larger than the maximum number of groups in which any user is a member. If you do not increase the limit, then searches to retrieve the set of all static groups in which the user is a member may be unindexed and therefore very expensive.

Summary of Commands to Search for Group Membership

The following summary of commands show the fastest way to retrieve direct or indirect member DNs for groups.

- To retrieve direct member (non-nested) DNs of group "`cn=group.1,ou=groups,dc=example,dc=com`".

  ```
  $ bin/ldapsearch --baseDN "cn=group.1,ou=Groups,dc=example,dc=com"
   "(objectClass=*)" uniqueMember member
  ```
- To retrieve direct member entries (non-nested) under "`dc=example,dc=com`" of group "`cn=group.1,ou=groups,dc=example,dc=com`". This is useful when attributes from member entries are used in the filter or being returned.

  ```
  $ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com"
   "(isDirectMemberOf=cn=group.1,ou=Groups,dc=example,dc=com)"
  ```
- To retrieve group DNs in which user "`uid=user.2,ou=people,dc=example,dc=com`" is a direct member (non-nested, static groups).

  ```
  $ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com"
   "(objectClass=*)" isDirectMemberOf
  ```

- To retrieve all member entries under `ou=people,dc=example,dc=com` of group "`cn=group.1,ou=groups,dc=example,dc=com`".

  ```
  $ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com"
   "(isMemberOf=cn=group.1,ou=Groups,dc=example,dc=com)"
  ```
- To retrieve the group DNs in which user "`uid=user.2,ou=people,dc=example,dc=com`" is a member.

  ```
  $ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com"
   "(objectClass=*)" isMemberOf
  ```

Migrating Sun/Oracle Groups

You can migrate Sun/Oracle static and dynamic groups to PingDirectory Server groups. The following sections outline the procedures for migrating static groups to both Ping Identity static groups and virtual static groups as well as how to migrate dynamic groups. For information about the differences in access control evaluation between Sun/Oracle and the PingDirectory Server, see Migrating ACIs from Sun/Oracle to PingDirectory Server.

Migrating Static Groups

The PingDirectory Server supports static LDAP groups with structural object classes of `groupOfNames`, `groupOfUniqueNames`, or `groupOfEntries`. In general, static groups may be imported without modification.

A FIFO entry cache can be enabled to cache group-to-user mappings, which improves performance when accessing very large entries, though at the expense of greater memory consumption. The PingDirectory Server provides an out-of-the-box FIFO entry cache object for this purpose. This object must be explicitly enabled using `dsconfig` as described in *Using the Entry Cache to Improve the Performace of Large Static Groups*.

To Migrate Static Groups

1. Run the `migrate-ldap-schema` tool to enumerate any schema differences between the DSEE deployment and the Ping Identity deployment.
2. Run the `migrate-sun-ds-config` tool to enumerate any configuration differences between the DSEE deployment and the Ping Identity deployment.
3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.
4. Import the existing users and groups using the `import-ldif` tool.
5. From the PingDirectory Server root directory, open the `sun-ds-compatibility.dsconfig` file in the `docs` folder using a text editor.
6. Find the **FIFO Entry Cache** section and, after reading the accompanying comments, enable the corresponding `dsconfig` command by removing the comment character ("#").

   ```
   $ bin/dsconfig set-entry-cache-prop \
     --cache-name "Static Group Entry Cache" --set enabled:true
   ```
7. Enable the Referential Integrity Plug-in. This will ensure that references to an entry are automatically updated when the entry is deleted or renamed.

   ```
   $ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" --set
    enabled:true
   ```

   If this Directory Server is part of a replication topology, you should enable the Referential Integrity Plug-in for each replica.

Migrating Static Groups to Virtual Static Groups

In many cases, electing to use virtual static groups in place of static groups can produce marked performance gains without any need to update client applications. The specifics of a migration to virtual static groups varies depending on the original DIT, but the general approach involves identifying common membership traits for all members of each group and then expressing those traits in the form of an LDAP URL.

In the following example, the common membership trait for all members of the All Users group is the parent DN ou=People,dc=example,dc=com. In other cases, a common attribute may need to be used. For example, groups based on the location of its members could use the `l` (location) or `st` (state) attribute.

To Migrate DSEE Static Groups to Virtual Static Groups

In the following example, consider the common case of an "All Users" group, which contains all entries under the parent DN "`ou=People,dc=example,dc=com`". When implemented as a virtual static group, this group may have a large membership set without incurring the overhead of a static group.

1. First, create a dynamic group.

```
dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Dynamic All Users
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(objectClass=person)
```

2. Next, create a virtual static group that references the dynamic group.

```
dn: cn=All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
objectClass: ds-virtual-static-group
cn: All Users
ds-target-group-dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
```

3. Finally, the `Virtual Static uniqueMember` virtual attribute must be enabled to populate the All Users group with `uniqueMember` virtual attributes.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
 uniqueMember" \
  --set enabled:true
```

4. Confirm that the virtual static group is correctly configured by checking a user's membership in the group.

```
$ bin/ldapsearch --baseDN "cn=All Users,ou=Groups,dc=example,dc=com" \
  --searchScope base "(uniqueMember=uid=user.0,ou=People,dc=example,dc=com)"
 1.1
```

```
dn: cn=All Users,ou=Groups,dc=example,dc=com
```

5. The ability to list all members of a virtual static group is disabled by default. You may enable this feature, but only if specifically required by a client application.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
 uniqueMember" \
  --set allow-retrieving-membership: true
```

Note: The virtual static group may also be implemented using the `groupOfNames` object class instead of `groupOfUniqueNames`. In that case, you must update the `Virtual Static member` configuration object instead of the `Virtual Static uniqueMember` configuration object.

Migrating Dynamic Groups

The PingDirectory Server supports dynamic groups with the `groupofURLs` object class. In general, dynamic groups may be imported without modification.

To Migrate Dynamic Groups

1. Run the `migrate-ldap-schema` tool to enumerate any schema differences between the DSEE deployment and the Ping Identity deployment.
2. Run the `migrate-sun-ds-config` tool to enumerate any configuration differences between the DSEE deployment and the Ping Identity deployment.
3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.
4. Import the existing users and groups using the `import-ldif` tool.

Chapter 14

Encrypting Sensitive Data

Topics:

The Directory Server provides several ways that you can protect sensitive information in the server. If not enabled during server setup, you can enable on-disk encryption for data in backends as well as in the changelog and the replication databases, and you can also protect sensitive attributes by limiting the ways that clients may interact with them.

This chapter presents the following topics:

Encrypting and Protecting Sensitive Data

The Directory Server provides an encryption-settings database that holds encryption and decryption definitions to protect sensitive data. You can enable on-disk encryption for data in backends as well as in the changelog and the replication databases. You can also protect sensitive attributes by limiting the ways that clients may interact with them.

About the Encryption-Settings Database

The encryption-settings database is a repository that the server uses to hold information for encrypting and decrypting data. The database contains any number of *encryption-settings definitions* that specifies information about the cipher transformation and encapsulates the key used for encryption and decryption.

Before data encryption can be enabled, you first need to create an encryption-settings definition. An encryption-settings definition specifies the cipher transformation that should be used to encrypt the data, and encapsulates the encryption key. The `encryption-settings` command-line tool can be used to manage the encryption settings database, including creating, deleting, exporting, and importing encryption-settings definitions, listing the available definitions, and indicating which definition should be used for subsequent encryption operations.

Although the encryption-settings database can have multiple encryption-settings definitions, only one of them can be designated as the *preferred* definition. The preferred encryption-settings definition is the one that will be used for any subsequent encryption operations. Any existing data that has not yet been encrypted remains unencrypted until it is rewritten (e.g., as a result of a modify or modify DN operation, or if the data is exported to LDIF and re-imported). Similarly, if you introduce a new preferred encryption-settings definition, then any existing encrypted data will continue to use the previous definition until it is rewritten. If you do change the preferred encryption-settings definition for the server, then it is important to retain the previous definitions until you are confident that no remaining data uses those older keys.

Supported Encryption Ciphers and Transformations

The set of encryption ciphers that are supported by the Directory Server is limited to those ciphers supported by the JVM in which the server is running. For specific reference information about the algorithms and transformations available in all compliant JVM implementations, see the following:

- Java Cryptography Architecture Reference Guide
- Java Cryptography Architecture Standard Algorithm Name Documentation

When configuring encryption, the cipher to be used must be specified using a key length (in bits) and either a cipher algorithm name (e.g., "AES") or a full cipher transformation which explicitly specifies the mode and padding to use for the encryption (e.g., "AES/CBC/ PKCS5Padding"). If only a cipher algorithm is given, then the default mode and padding for that algorithm will be automatically selected.

The following cipher algorithms and key lengths have been tested using the Sun/Oracle JVM:

Table 26: Cipher Algorithms

Cipher Algorithm	Key Length (bits)
AES	128
Blowfish	128
DES	64
DESede	192
RC4	128

Note: By default, some JVM implementations may come with limited encryption strength, which may restrict the key lengths that can be used. For example, the Sun/Oracle JVM does not allow AES with 192-bit or 256-bit keys unless the unlimited encryption strength policy files are downloaded and installed.

The Directory Server supports four Cipher Stream Providers, which are used to obtain cipher input and output streams to read and write encrypted data.

Table 27: Cipher Stream Providers

Cipher Stream Providers	Description
Default	Default cipher stream provider using a hard-coded default key.
File-Based	Used to read a specified file in order to obtain a password used to generate cipher streams for reading and writing encrypted data.
Third-Party	Used to provide cipher stream provider implementations created in third-party code using the Server SDK.
Wait-for-Passphrase	Causes the server to wait for an administrator to enter a passphrase that will be used to derive the key for cipher streams. You can supply the passphrase to the server by running `encryption-settings supply-passphrase`.

Using the encryptions-settings Tool

The `encryption-settings` tool provides a mechanism for interacting with the server's encryption-settings database. It may be used to list the available definitions, create new definitions, delete existing definitions, and indicate which definition should be the preferred definition. It may also be used to export definitions to a file for backup purposes and to allow them to be imported for use in other Directory Server instances.

To List the Available Encryption Definitions

- Use the `encryption-settings` tool with the `list` subcommand to display the set of available encryption settings definitions. This subcommand does not take any arguments. For each definition, it will include the unique identifier for the definition, as well as the cipher transformation and key length that will be used for encryption and whether it is the preferred definition.

```
$ bin/encryption-settings list
```

```
Encryption Settings Definition ID: 4D86C7922F71BB57B8B5695D2993059A26B8FC01
Preferred for New Encryption: false
Cipher Transformation: DESede
Key Length (bits): 192

Encryption Settings Definition ID: F635E109A8549651025D01D9A6A90F7C9017C66D
Preferred for New Encryption: true
Cipher Transformation: AES
Key Length (bits): 128
```

Creating Encryption-Settings Definitions

To create a new encryption-settings definition, use the create subcommand. This subcommand takes the following arguments:

- **--cipher-algorithm {algorithm}**. Specifies the base cipher algorithm that should be used. This should just be the name of the algorithm (e.g., "AES", "DES", "DESede", "Blowfish", "RC4", etc.). This argument is required.
- **--cipher-transformation {transformation}**. Specifies the full cipher transformation that should be used, including the cipher mode and padding algorithms (e.g., "AES/CBC/ PKCS5Padding"). This argument is optional, and if it is not provided, then the JVM-default transformation will be used for the specified cipher algorithm.

- **--key-length-bits {length}**. Specifies the length of the encryption key in bits (e.g., 128). This argument is required.
- **--set-preferred**. Indicates that the new encryption-settings definition should be made the preferred definition and therefore should be used for subsequent encryption operations in the server. When creating the first definition in the encryption-settings database, it will automatically be made the preferred definition.

To Create an Encryption-Settings Definition

- Use the `encryption-settings` tool with the create subcommand to specify the definition.

```
$ bin/encryption-settings create --cipher-algorithm AES \
  --key-length-bits 128 --set-preferred
```

```
Successfully created a new encryption settings definition with ID
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Changing the Preferred Encryption-Settings Definition

To change the preferred encryption-settings definition, use the `encryption-settings` tool with the `set-preferred` subcommand. This subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument is required.

To Change the Preferred Encryption-Settings Definition

- Use the `encryption-settings` tool with the set-preferred subcommand to change a definition to a preferred definition.

```
$ bin/encryption-settings set-preferred --id
 4D86C7922F71BB57B8B5695D2993059A26B8FC01
```

```
Encryption settings definition 4D86C7922F71BB57B8B5695D2993059A26B8FC01 was
successfully set as the preferred definition for subsequent encryption
 operations
```

Deleting an Encryption-Settings Definition

To delete an encryption-settings definition, use the `encryption-settings` tool with the delete subcommand. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be deleted. This argument is required.

Never delete an encryption-settings definition if data in the server is still encrypted using the settings contained in that definition. Any data still encrypted with a definition that has been removed from the database will be inaccessible to the server and will cause errors for any attempt to access it. This includes the replicationChanges and changelog databases, which the `re-encode-entries` tool will not re-encode with the new encryption-settings definition. Therefore, wait for the amount of time defined in the `replication-purge-delay`, of the Replication Server, and `changelog-maximum-age` of the changelog Backend (if enabled) before removing previous encryption-settings definitions. To safely delete a compromised encryption-settings definition, see the the *Dealing with a Compromised EncryptionKey* section.

To stop using a definition for encryption and use a different definition, make sure that the desired definition exists in the encryption-settings database and set it to be the preferred definition. As long as the encryption key has not been compromised, there is no harm in having old encryption-settings definitions available to the server, and it is recommended that they be retained just in case they are referenced by something.

The preferred encryption-settings definition cannot be deleted unless it is the only one left. To delete the currently-preferred definition when one or more other definitions are available, make one of the other definitions preferred as described in the previous section.

To Delete an Encryption-Settings Definition

- Use the `encryption-settings` command with the delete subcommand. Make sure to include the --id argument to specify the definition.

```
$ bin/encryption-settings delete --id
 F635E109A8549651025D01D9A6A90F7C9017C66D
```

```
Successfully deleted encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Configuring the Encryption-Settings Database

Because the encryption-settings database contains the encryption keys used to protect server data, the contents of the encryption-settings database is itself encrypted. By default, the server will derive a key to use for this purpose, but it is recommended that you customize the logic used to access the encryption-settings database with a cipher stream provider. The Server SDK provides an API that can be used to create custom cipher stream provider implementations, but the server also comes with one that will obtain the key from a PIN file that you create (see the example procedure below).

To Configure the Encryption-Settings Database

1. Use `dsconfig` to configure the server so that the encryption-settings database is encrypted with a PIN contained in the file `config/encryption-settings.pin`.

```
$ bin/dsconfig create-cipher-stream-provider \
  --provider-name "Encryption Settings PIN File" \
  --type file-based \
  --set enabled:true \
  --set password-file:config/encryption-settings.pin
```

2. Use `dsconfig` to set the global configuration property for the cipher stream provider, which sets the on-disk encryption.

```
$ bin/dsconfig set-global-configuration-prop \
  --set "encryption-settings-cipher-stream-provider:Encryption Settings PIN
 File"
```

3. Use the `encryption-settings` tool to create a new encryption-settings definition. This command automatically generates a new 256-bit encryption key for use with AES encryption, and mark it as the preferred definition for future encryption operations in the server. Note that this command will fail if you do not have the unlimited encryption strength policy installed as described in the previous section (if you do not have that policy installed, then you are restricted to a 128-bit key for AES encryption).

```
$ bin/encryption-settings create \
  --cipher-algorithm AES \
  --key-length-bits 256 \
  --set-preferred
```

4. Obtain a list of the definitions in the encryption-settings database.

```
$ bin/encryption-settings list
```

5. You can export an encryption-settings definition from the database using a command like the following where the encryption-settings ID should be changed as necessary to suit your deployment:

```
$ bin/encryption-settings export \
  --id DA39A3EE5E6B4B0D3255BFEF95601890AFD80709 \
  --output-file /tmp/exported-key \
  --pin-file /tmp/exported-key.pin
```

6. If no PIN file is specified, then you will be interactively prompted to provide it. To import an encryption-settings definition into the database on another server.

```
$ bin/encryption-settings import \
  --input-file /tmp/exported-key \
  --pin-file /tmp/exported-key.pin \
  --set-preferred
```

Backing Up and Restoring the Encryption-Settings Definitions

If using data encryption in a Directory Server instance, do not lose the encryption-settings definitions used to encrypt data in the server. If an encryption-settings definition is lost, any data encrypted with that definition will be completely inaccessible. Make sure the encryption-settings definitions are backed up regularly.

The Directory Server provides two different mechanisms for backing up and restoring encryption-settings definitions. One or more encryption-settings definitions can be exported and imported using the `encryption-settings` tool. Or, the entire encryption-settings database can be backed up and restored using the Directory Server's `backup` and `restore` tools.

If a pin file is used to define a passphrase to the encryption-settings database, the passphrase must be backed up and kept secure independently of the userRoot and encryption-settings database backups. The passphrase in the pin file is needed if the encryption-settings database is to be restored into a different server root.

Exporting Encryption-Settings Definitions

To back up an individual definition (or to export it from one server so that you can import it into another), use the export subcommand to the `encryption-settings` command. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument can be specified multiple times. If it is omitted, all definitions are exported.
- **--output-file {path}**. Specifies the path to the output file to which the encryption-settings definition will be written. This argument is required.
- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.

To Export an Encryption-Settings Definition

- Use the `encryption-settings` tool with the export subcommand to export the definition to a file.

```
$ bin/encryption-settings export --id
 F635E109A8549651025D01D9A6A90F7C9017C66D \
  --output-file /tmp/exported-key
Enter the PIN to use to encrypt the definition:
Re-enter the encryption PIN:
```

```
Successfully exported encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D to file /tmp/exported-key
```

Importing Encryption-Settings Definitions

To import an encryption-settings definition that has been previously exported, use the `encryption-settings` tool with the import subcommand. The subcommand takes the following arguments:

- **--input-file {path}**. Specifies the path to the file containing the exported encryption-settings definition. This argument is required.
- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.
- **--set-preferred**. Specifies that the newly-imported encryption-settings definition should be made for the preferred definition for subsequent encryption-settings.

To Import an Encryption-Settings Definition

- Use the `encryption-settings` tool with the import subcommand to import the definition to a file.

```
$ bin/encryption-settings import --input-file /tmp/exported-key --set-
preferred
Enter the PIN used to encrypt the definition:
```

```
Successfully imported encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D from file /tmp/exported-key
```

Enabling Data Encryption in the Server

Encryption can be enabled during server setup, by defining an encryption key and passphrase. This configuration should be used across all servers in a topology. On legacy systems or post setup, data encryption can be configured by having at least one encryption-settings definition available for use. Then, set the value of the `encrypt-data` global configuration property to true.

Setting the global configuration property will automatically enable data encryption for all types of backends that support it (including the changelog backend and indexes), as well as for the replication server database. All subsequent write operations will cause the corresponding records written into any of these locations to be encrypted. Any existing data will remain unencrypted until it is rewritten by a write operation. If you wish to have existing data encrypted, then you will need to export that data to LDIF and re-import it. This will work for both the data backends, the changelog, and indexes, but it is not an option for the replication database, so existing change records will remain unencrypted until they are purged. If this is not considered acceptable in your environment, then follow the steps in the *Dealing with a Compromised Encryption Key* to safely purge the replication database.

Configuration for backups and LDIF exports can be done with the following global properties:

- `automatically-compress-encrypted-ldif-exports`. Indicates whether to automatically compress LDIF exports that are also encrypted. If set to true, any LDIF export that is encrypted (either explicitly with `--encryptLDIF`,or implicitly with the `encrypt-ldif-exports-by-default` configuration property) will automatically be gzip-compressed. If this is false, encrypted LDIF exports can still be manually compressed using the `--compress` command-line argument.
- `backup-encryption-settings-definition-id`. The unique identifier for the encryption settings definition to use to generate the encryption key for encrypted backups by default. If this property is given a value, then a definition with that ID must exist in the server's encryption settings database. If this property is not given a value, but the server is configured with at least one encryption settings definition, then the preferred definition is used. If no encryption settings definitions are available, the server will use an internal key shared among servers in the topology. Regardless of this property's value, it can be overridden with the `backup` command-line tool. Providing one of the `--promptForEncryptionPassphrase` or `--encryptionPassphraseFile` arguments will generate the encryption key from the provided passphrase. Or, the `--encryptionSettingsDefinitionID` argument can be used to generate the key from the specified encryption settings definition.
- `encrypt-backups-by-default`. Indicates whether the server should encrypt backups by default. If set to true, a defined `backup-encryption-settings-definition-id` value will be used to generate the encryption key for the backup. If this property is true, and if a `backup-encryption-settings-definition-id` value is not specified, the server will try to use the preferred encryption settings definition to generate the encryption key. If the server is not configured with any encryption settings definitions, an internal key that is shared among instances in the topology is used. Regardless of this property's value, it can be overridden with the `backup` command-line tool's `--encrypt` argument, even if this property is set to false. The `--doNotEncrypt` argument will always cause the backup to be unencrypted, even if this property has a value of true.
- `encrypt-ldif-exports-by-default`. Indicates whether the server should encrypt LDIF exports by default. If set to true, and an `ldif-export-encryption-settings-definition-id` value is specified, then that encryption settings definition is used to generate the encryption key for the export. If this property is true, and an `ldif-export-encryption-settings-definition-id` value is not specified, the server will first try to use the preferred encryption settings definition to generate the encryption key. If the server is not configured with any encryption settings definitions, then an internal key that is shared among

instances in the topology is used. Regardless of this property's value, the default behavior can be overridden with the `export-ldif` command-line tool. The tool's `--encryptLDIF` argument will always encrypt the export, and the `--doNotEncryptLDIF` argument will always create an unencrypted export.

To Enable Data Encryption in the Server

- Use `dsconfig` to set the global configuration property for data encryption to true.

```
$ bin/dsconfig set-global-configuration-prop --set encrypt-data:true
```

Using Data Encryption in a Replicated Environment

Data encryption is only used for the on-disk storage for data within the server. Whenever clients access that data, it is presented in unencrypted form (although the communication with those clients may itself be encrypted using SSL or StartTLS). Replication, the communication of updates between replication servers, is always encrypted using SSL. Each server may apply data encryption in a completely independent manner and have different sets of encryption-settings definitions. It is also possible to have a replication topology containing some servers with data encryption enabled and others with it disabled.

However, when initializing the backend of one server from another server with data encryption enabled, then the server being initialized must have access to all encryption-settings definitions that may have been used for data contained in that backend. To do this, perform an export of the encryption-settings database on the source server using `bin/encryption-settings export` and import it on the target server using `bin/encryption-settings import`.

Dealing with a Compromised Encryption Key

If an encryption-settings definition becomes compromised such that an unauthorized individual obtains access to the encryption key, then any data encrypted with that definition is also vulnerable because it can be decrypted using that key. It is very important that the encryption-settings database be protected (e.g., using file permissions and/or filesystem ACLs) to ensure that its contents remain secure.

In the event that an encryption-settings definition is compromised, then you should immediately stop using that definition. Any data encrypted with the compromised key should be re-encrypted with a new definition or purged from the server. This can be done on one server at a time to avoid an environment-wide downtime, but it should be completed as quickly as possible on all servers that had used that definition at any point in the past in order to minimize the risk of that data becoming exposed.

To Deal with a Compromised Encryption Key

The recommended process for responding to a compromised encryption settings definition is as follows:

1. Create a new encryption-settings definition and make it the preferred definition for new writes.
2. Ensure that client traffic is routed away from the server instance to be updated. For example, if the Directory Server is accessed through a Directory Proxy Server, then you may set the `health-check-state` configuration property for any LDAP external server definitions that reference that server to have a value of unavailable.
3. Ensure that external clients are not allowed to write operations in the directory server instance. This may be accomplished by setting the `writability-mode` global configuration property to have a value of `internal-only`.
4. Wait for all outstanding local changes to be replicated out to other servers. This can be accomplished by looking at the monitor entries with the `ds-replication-server-handler-monitor-entry` object class to ensure that the value of the `update-sent` attribute is no longer increasing.
5. Stop the directory server instance.
6. Delete the replication server database by removing all files in the `changeLogDb` directory below the server root. As long as all local changes have been replicated out to other servers, this will not result in any data loss in the replication environment.

7. Export the contents of all local DB and changelog backends to LDIF. Then, re-import the data from LDIF, which will cause it to be encrypted using the new preferred encryption settings definition.
8. Export the compromised key from the encryption settings database to back it up in case it may be needed again in the future (e.g., if some remaining data was later found to have been encrypted with the key contained in that definition). Then, delete it from the encryption settings database so that it can no longer be used by that directory server instance.
9. Start the directory server instance.
10. Allow replication to bring the server back up-to-date with any changes processed while it was offline.
11. Re-allow externally-initiated write operations by changing the value of the global `writability-mode` configuration property back to enabled.
12. Re-configure the environment to allow client traffic to again be routed to that server instance (e.g., by changing the value of the "health-check-state" property in the corresponding LDAP external instance definitions in the Directory Proxy Server instances back to "dynamically-determined").

Configuring Sensitive Attributes

Data encryption is only applied to the on-disk storage for a Directory Server instance. It does not automatically protect information accessed or replicated between servers, although the server offers other mechanisms to provide that protection (i.e., SSL, StartTLS, SASL). Ensuring that all client communication uses either SSL or StartTLS encryption and ensuring that all replication traffic uses SSL encryption ensures that the data is protected from unauthorized individuals who may be able to eavesdrop on network communication. This communication security may be enabled independently of data encryption (although if data encryption is enabled, then it is strongly recommended that secure communication be used to protect network access to that data).

However, for client data access, it may not be as simple as merely enabling secure communication. In some cases, it may be desirable to allow insecure access to some data. In other cases, it may be useful to have additional levels of protection in place to ensure that some attributes are even more carefully protected. These kinds of protection may be achieved using sensitive attribute definitions.

Each sensitive attribute definition contains a number of configuration properties, including:

- **`attribute-type`**. Specifies the set of attribute types whose values may be considered sensitive. At least one attribute type must be provided, and all specified attribute types must be defined in the server schema.
- **`include-default-sensitive-operational-attributes`**. Indicates whether the set of sensitive attributes should automatically be updated to include any operational attributes maintained by the Directory Server itself that may contain sensitive information. At present, this includes the `ds-sync-hist` operation attribute, which is used for data required for replication conflict resolution and may contain values from other attributes in the entry.
- **`allow-in-filter`**. Indicates whether sensitive attributes may be used in filters. This applies not only to the filter used in search requests, but also filters that may be used in other places, like the assertion and join request controls. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be used in filters over both secure and insecure connections)
  - `Reject` (reject any request which includes a filter targeting one or more sensitive attributes over both secure and insecure connections)
  - `Secure-only` (allow sensitive attributes to be used in filters over secure connections, but reject any such requests over insecure connections)
- **`allow-in-add`**. Indicates whether sensitive attributes may be included in entries created by LDAP add operations. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be included in add requests over both secure and insecure connections)
  - `Reject` (reject any add request containing sensitive attributes over both secure and insecure connections)
  - `Secure-only` (allow sensitive attributes to be included in add requests received over a secure connection, but reject any such requests over an insecure connection)

- **`allow-in-compare`**. Indicates whether sensitive attributes may be targeted by the assertion used in a compare operation. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be targeted by requests over both secure and insecure connections)
  - `Reject` (reject any compare request targeting a sensitive attribute over both secure and insecure connections)
  - `Secure-only` (allow compare requests targeting sensitive attributes over a secure connection, but reject any such requests over an insecure connection)
- **`allow-in-modify`**. Indicates whether sensitive attributes may be updated using modify operations. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be modified by requests over both secure and insecure connections)
  - `Reject` (reject any modify request updating a sensitive attribute over both secure and insecure connections)
  - `Secure-only` (only modify requests updating sensitive attributes over a secure connection, but reject any such request over an insecure connection)

The `allow-in-returned-entries`, `allow-in-filter`, `allow-in-add`, `allow-in-compare`, and `allow-in-modify` properties all have default values of `secure-only`, which prevents the possibility of exposing sensitive data in the clear to anyone able to observe network communication.

If a client connection policy references a sensitive attribute definition, then any restrictions imposed by that definition will be enforced for any clients associated with that client connection policy. If multiple sensitive attribute definitions are associated with a client connection policy, then the server will use the most restrictive combination of all of those sets.

Note that sensitive attribute definitions work in conjunction with other security mechanisms defined in the server and may only be used to enforce additional restrictions on clients. Sensitive attribute definitions may never be used to grant a client additional access to information that it would not have already had through other means. For example, if the `employeeSSN` attribute is declared to be a sensitive attribute and the `allow-in-returned-entries` property has a value of `Secure-only`, then the `employeeSSN` attribute will only be returned to those clients that have both been granted permission by the access control rules defined in the server and are communicating with the server over a secure connection. The `employeeSSN` attribute will be stripped out of entries returned to clients normally authorized to see it if they are using insecure connections, and it will also be stripped out of entries for clients normally not authorized to see it even if they have established secure connections.

To Create a Sensitive Attribute

1. To create a sensitive attribute, you must first create one or more sensitive attribute definitions.

   For example, to create a sensitive attribute definition that will only allow access to the `employeeSSN` attribute by clients using secure connections, the following configuration changes may be made:

   ```
   $ bin/dsconfig create-sensitive-attribute \
     --attribute-name "Employee Social Security Numbers" \
     --set attribute-type:employeeSSN \
     --set include-default-sensitive-operational-attributes:true \
     --set allow-in-returned-entries:secure-only \
     --set allow-in-filter:secure-only \
     --set allow-in-add:secure-only \
     --set allow-in-compare:secure-only \
     --set allow-in-modify:secure-only
   ```
2. Associate those sensitive attribute definitions with the client connection policies for which you want them to be enforced.

   ```
   $ bin/dsconfig set-client-connection-policy-prop --policy-name default \
     --set "sensitive-attribute:Employee Social Security Numbers"
   ```

Configuring Global Sensitive Attributes

Administrators can assign one or more sensitive attribute definitions to a client connection policy. However, in an environment with multiple client connection policies, it could be easy to add a sensitive attribute definition to one policy but overlook it in another. The Directory Server supports the ability to define sensitive attributes as a global configuration option so that they will automatically be used across all client connection policies.

To Configure a Global Sensitive Attribute

- Run `dsconfig` to add a global sensitive attribute across all client connection policies. The following command adds the `employeeSSN` as a global sensitive attribute, which is applied across all client connection policies.

```
$ bin/dsconfig set-global-configuration-prop --add "sensitive-
attribute:employeeSSN"
```

Excluding a Global Sensitive Attribute on a Client Connection Policy

Administrators can set a global sensitive attribute across all client connection policies. However, there may be cases when a specific directory server must exclude the sensitive attribute as it may not be needed for client connection requests. For example, in most environments it is good to declare the `userPassword` attribute to be a sensitive attribute in a manner that prevents it from being read by external clients. Further, this solution is more secure than protecting the `password` attribute using the server's default global ACI, which only exists for backwards compatibility purposes. If the Data Sync Server is installed, then it does need to be able to access passwords for synchronization purposes. In this case, the administrator can set `userPassword` to be a sensitive attribute in all client connection policies, but exclude it in a policy specifically created for use by the Data Sync Server. The Directory Server provides an `exclude-global-sensitive-attribute` property for this purpose.

To Exclude a Global Sensitive Attribute on a Client Connection Policy

1. Run `dsconfig` to remove the global ACI that limits access to the `userPassword` or `authPassword` attribute. This is present for backwards compatibility.

```
$ bin/dsconfig set-access-control-handler-prop \
  --remove 'global-aci:(targetattr="userPassword || authPassword")
   (version 3.0; acl "Prevent clients from retrieving passwords from the
 server";
  deny (read,search,compare) userdn="ldap:///anyone";)'
```

2. Run `dsconfig` to add the `userPassword` attribute as a global sensitive attribute, which is applied to all client connection policies. Do this by adding the built-in "Sensitive Password Attributes" Sensitive Attribute definition to the Global Configuration.

```
$ bin/dsconfig set-global-configuration-prop \
  --add "sensitive-attribute:Sensitive Password Attributes"
```

3. If the server is designated to synchronize passwords with a Sync Server, then it is necessary to configure a client connection policy for the Sync User to exclude the global sensitive attribute. The following is an example on how to create a new policy if the Data Sync Server binds with the default DN of `cn=Sync User,cn=Root DNs,cn=config`.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Requests by Sync Users" \
  --type simple \
  --set user-auth-type:internal \
  --set user-auth-type:sasl \
  --set user-auth-type:simple \
  --set "included-user-base-dn:cn=Sync User,cn=Root DNs,cn=config"
```

```
$ bin/dsconfig create-client-connection-policy \
  --policy-name "Data Sync Server Connection Policy" \
  --set enabled:true \
  --set evaluation-order-index:9998 \
  --set "connection-criteria:Requests by Sync Users" \
  --set "exclude-global-sensitive-attribute:Sensitive Password Attributes"
```

Chapter

15

Working with the LDAP Changelog

Topics:

The Directory Server provides a client-accessible LDAP changelog (based on the Changelog Internet Draft Specification) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services.

This chapter will present the following topics related to the LDAP Changelog:

Overview of the LDAP Changelog

The Directory Server provides a client-accessible LDAP changelog (based on the Changelog Internet Draft Specification) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services. For example, you can install the Data Sync Server that monitors the LDAP changelog for any updates that occur on a source directory server and synchronizes these changes to a target DIT or database server. The Directory Server provides an additional feature in that the LDAP changelog supports virtual attributes.

Note: The LDAP Changelog should not be confused with the Replication Changelog. The main distinction is as follows:

- The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at `<server-root>/db/changelog`.
- The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at `<server-root>/changelogDB`.

Key Changelog Features

As of version 3.2, the Directory Server supports two new Changelog Backend properties that allow access control filtering and sensitive attribute evaluation for targeted entries. External client applications can change the contents of attributes they can see in the targeted entry based on the access control rules applied to the associated base DN.

- **apply-access-controls-to-changelog-entry-contents**. Indicates whether the contents of changelog entry attributes (i.e., `changes`, `deletedEntryAttrs`, `ds-changelog-entry-key-attr-values`, `ds-changelog-before-values`, and `ds-changelog-after-values`) are subject to access control and/ or sensitive attribute evaluation to limit data that LDAP clients can see. The client must have the access control permissions to read changelog entries to retrieve them in any form. If this feature is enabled and the client does not have permission to read an entry at all, or if that client does not have permission to see any attributes that were targeted by the change, then the associated changelog entries targeted by those operations will be suppressed. If a client does not have permission to see certain attributes within the target entry, then references to those attributes in the changelog entry will also be suppressed. This property only applies to standard LDAP searches of the `cn=changelog` branch.
- **report-excluded-changelog-attributes**. Indicates whether to include additional information about any attributes that may have been removed due to access control filtering. This property only applies to content removed as a result of processing performed by the `apply-access-controls-to-changelog-entry-contents` property. Possible values are:
  - **none** - Indicates that changelog entries should not include any information about attributes that have been removed.
  - **attribute-counts** - Indicates that changelog entries should include a count of user and/ or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, the number of the excluded user attributes will be reported in the `ds-changelog-num-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the number of the excluded operational attributes will be reported in the `ds-changelog-num-excluded-operational-attributes` attribute of the changelog entry. Both the `ds-changelog-num-excluded-user-attributes` and `ds-changelog-num-excluded-operational-attributes` are operational and must be explicitly requested by clients (or all operational attributes requested using "+") to be returned.
  - **attribute-names** - Indicates that changelog entries should include the names of user and/or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, then the names of the excluded user attributes will be reported in the `ds-changelog-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the names of the excluded operational attributes will be reported in the `ds-change-log-excluded-operational-attribute` attribute of the changelog entry. Both the `ds-changelog-excluded-user-attribute` and `ds-changelog-excluded-operational-`

`attribute` attributes are operational and must be explicitly requested by clients (or all operational attributes requested via "+") to be returned.

To Enable Access Control Filtering in the LDAP Changelog

To set up access control to the LDAP Changelog, use the `dsconfig` tool to enable the properties to the Changelog Backend. Only admin users with the `bypass-acl` privilege can read the changelog.

1. Enable the `apply-access-control-to-changelog-entry-contents` property to allow LDAP clients to undergo access control filtering using standard LDAP searches of the `cn=changelog` backend.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "apply-access-controls-to-changelog-entry-contents:true"
```

   Access control filtering will be applied regardless of the value of the `apply-access-controls-to-changelog-entry-contents` setting when the Changelog Backend is servicing requests from an PingDataSync Server that has the `filter-changes-by-user` Sync Pipe property set.

2. Optional. Set the `report-excluded-changelog-attributes` property to include a count of users that have been removed through access control filtering. The count appears in the `ds-changelog-num-excluded-user-attributes` attribute for users and the `ds-changelog-num-excluded-operational-attributes` attribute for operational attributes.

```
 $ bin/dsconfig set-backend-prop --backend-name "changelog" \
 --set "report-excluded-changelog-attributes:attribute-counts"
```

Useful Changelog Features

The Directory Server provides two useful changelog configuration properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

- **changelog-max-before-after-values**. Setting this property to a non-zero value causes all of the old values and all of the new values (up to the specified maximum) for each changed attribute to be stored in the changelog entry. The values will be stored in the `ds-change-log-before-values` and `ds-changelog-after-values` attributes on the changelog entry. These attributes are not present by default.

  Note: The `changelog-max-before-after-values` property can be expensive for attributes with hundreds or thousands of values, such as a group entry.

  If any attribute has more than the maximum number of values, their names and number of before/after values will be stored in the `ds-changelog-attr-exceeded-max-values-count` attribute on the changelog entry. This is a multi-valued attribute whose format is:

```
attr=attributeName,beforeCount=100,afterCount=101
```

  where "attributeName" is the name of the attribute and the "beforeCount" and "afterCount" are the total number of values for that attribute before and after the change, respectively. This attribute indicates that you need to reset the `changelog-max-before-after-values` property to a higher value. When this attribute is set, an alert will be generated.

  Note: If the number of values for an attribute exceeds the maximum value set by the `changelog-max-before-after-values` property, then those values will not be stored.

- **changelog-include-key-attribute**. This property is used for correlation attributes that need to be synchronized across servers, such as `uid`. It causes the current (after-change) value of the specified attributes to be recorded in the `ds-changelog-entry-key-attr-values` attribute on the changelog entry. This applies for all change types. On a DELETE operation, the values are from the entry before they were deleted.

  The key values will be recorded on every change and override the settings configured in `changelog-include-attribute`, `changelog-exclude-attribute`, `changelog-deleted-entry-include-attribute`, or `changelog-deleted-entry-exclude-attribute`.

Example of the Changelog Features

After the `changelog-max-before-after-values` property is set, the before-and-after values of any change attribute will be recorded in the LDAP Changelog. For example, given a simple entry with two multi-valued mail attributes:

```
dn: uid=test,dc=example,dc=com
objectclass: inetorgperson
cn: test user
sn: user
description: oldDescription
mail: test@yahoo.com
mail: test@gmail.com
```

Then, apply the following changes to the entry:

```
dn: uid=test,dc=example,dc=com
changetype: modify
add: mail
mail: test@hotmail.com
-
delete: mail
mail: test@yahoo.com
-
replace: description
description: newDescription
```

The resulting changelog would record the following attribute values:

```
dn: changeNumber=1,cn=changelog
objectClass: top
objectClass: changeLogEntry
targetDN: uid=test,dc=example,dc=com
changeType: modify
changes::
YWRkOiBtYWlsCm1haWw6IHRlc3RAaG90bWFpbC5jb20KLQpkZWxldGU6IG1haWwKbWFpbDogdGVzdEB5YWh
vby5jb20KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogbmV3RGVzY3JpcHRpb24KLQpyZX
BsYWNlOiBtb2RpZmllcnNOYW1lCm1vZGlmaWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VyLGNuPVJvb
3QgRE5zLGNuPWNvbmZpZwotCnJlcGxhY2U6IGRzLXVwZGF0ZS10aW1lCmRzLXVwZGF0ZS10aW1lOjogQUFB
QkxxQitIaTQ9Ci0KAA==
ds-changelog-before-values::
 ZGVzY3JpcHRpb246IG9sZERlc2NyaXB0aW9uCm1haWw6IHRlc3RAeW
Fob28uY29tCm1haWw6IHRlc3RAZ21haWwuY29tCmRzLXVwZGF0ZS10aW1lOjogQUFBQkxxQjdaZ1E9Cm1vZ
GlmaWVyc05hbWU6IGNuPURpcmVjdG 9yeSBNYW5hZ2VyLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwo=
ds-changelog-after-values::
 ZGVzY3JpcHRpb246IG5ld0Rlc2NyaXB0aW9uCm1haWw6IHRlc3RAZ21
haWwuY29tCm1haWw6IHRlc3RAaG90bWFpbC5jb20KZHMtdXBkYXRlLXRpbWU6OiBBQUFCTHFCK0hpND0KbW
9kaWZpZXJzTmFtZTogY249RGlyZWN0b3J5IE1hbmFnZXIsY249Um9vdCBETnMsY249Y29uZmlnCg==
ds-changelog-entry-key-attr-values:: dWlkOiB0ZXN0Cg==
changenumber: 1
```

You can run the `bin/base64` decode -d command-line tool to view the decoded value for the changes, `ds-changelog-before-values`, `ds-changelog-after-values` attributes:

After base64 decoding, the `changes` attribute reads:

```
add: mail
mail: test@hotmail.com
-
delete: mail
mail: test@yahoo.com
-
replace: description
```

```
description: newDescription
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
-
```

After base64 decoding, the `ds-changelog-before-values` attribute reads:

```
description: oldDescription
mail: test@yahoo.com
mail: test@gmail.com
modifyTimestamp: 20131010020345.546Z
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

After base64 decoding, the `ds-changelog-after-values` attribute reads:

```
description: newDescription
mail: test@gmail.com
mail: test@hotmail.com
modifyTimestamp: 20131010020345.546Z
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

Viewing the LDAP Changelog Properties

You can view the LDAP changelog properties by running the `dsconfig` get-backend-prop command and specifying the changelog backend.

To View the LDAP Changelog Properties Using dsconfig Non-Interactive Mode

- Use `dsconfig` to view the changelog properties on the Directory Server. To view "advanced" properties that are normally hidden, add the --advanced option when running the command. For a specific description of each property, see the *PingDirectory Server Configuration Reference*.

```
$ bin/dsconfig get-backend-prop --backend-name changelog
```

```
Property                                     : Value(s)
--------------------------------------------:-------------
backend-id                                   : changelog
description                                  : -
enabled                                      : false
writability-mode                             : disabled
base-dn                                      : cn=changelog
set-degraded-alert-when-disabled             : false
return-unavailable-when-disabled             : false
db-directory                                 : db
db-directory-permissions                     : 700
changelog-maximum-age                        : 2 d
db-cache-percent                             : 1
changelog-include-attribute                  : -
changelog-exclude-attribute                  : -
changelog-deleted-entry-include-attribute    : -
changelog-deleted-entry-exclude-attribute    : -
changelog-include-key-attribute              : -
changelog-max-before-after-values            : 0
changelog-write-batch-size                   : 100
changelog-purge-batch-size                   : 1000
changelog-write-queue-capacity               : 100
write-lastmod-attributes                     : true
```

```
use-reversible-form                        : false
je-property                                : -
```

Enabling the LDAP Changelog

By default, the LDAP changelog is disabled on the Directory Server. If you are using the `dsconfig` tool in interactive mode, the changelog appears in the Backend configuration as a Standard object menu item.

Note: You can enable the feature using the `dsconfig` tool only if required as it can significantly affect LDAP update performance.

To Enable the LDAP Changelog Using dsconfig Non-Interactive Mode

- Use `dsconfig` to enable the changelog property on the Directory Server.

```
$ bin/dsconfig set-backend-prop \
  --backend-name changelog --set enabled:true
```

To Enable the LDAP Changelog Using Interactive Mode

1. Use `dsconfig` to enable the changelog on each server in the network. Then, authenticate to the server by entering the host name, LDAP connection, port, bindDN and bind password.

```
$ bin/dsconfig
```

2. On the Directory Server main menu, type `o` to change from the Basic object level to the Standard object level.
3. Enter the option to select the Standard Object level.
4. On the Directory Server main menu, type the number corresponding to Backend.
5. On the **Backend Management** menu, enter the option to view and edit an existing backend.
6. Next, you should see a list of the accessible backends on your system. For example, you may see options for the `changelog` and `userRoot` backends. Enter the option to work with the changelog backend.
7. On the **Changelog Backend properties** menu, type the number corresponding to the Enabled property.
8. On the **Enabled Property** menu, type the number to change the `Enabled` property to TRUE.
9. On the **Backend Properties** menu, type `f` to apply the change. If you set up the server in a server group, type `g` to update all of the servers in the group. Otherwise, repeat steps 1-10 on the other servers.
10. Verify that changes made to the data are recorded in the changelog.

Changing the LDAP Changelog Database Location

In cases where disk space issues arise, you can change the on-disk location of the LDAP Change Log database. The changelog backend supports a `db-directory` property that specifies the absolute or relative path (i.e., relative to the local server root) to the filesystem directory that is used to hold the Oracle Berkeley DB Java Edition database files containing the data for this backend.

If you change the changelog database location, you must stop and then restart the Directory Server for the change to take effect. If the changelog backend is already enabled, then the database files must be manually moved or copied to the new location while the server is stopped.

To Change the LDAP Changelog Location Using dsconfig Non-Interactive Mode

1. Use `dsconfig` to change the database location for the LDAP Changelog, which by default is at `<server-root>/db`. The following command sets the LDAP changelog backend to `<server-root>/db2`. Remember to include the LDAP connection parameters (e.g., hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "db-directory:db2" --set "enabled:true"
```

The database files are stored under `<server-root>/db2/changelog`. The files for this backend are stored in a sub-directory named after the `backend-id` property.

2. Stop and restart the server. Since the LDAP changelog backend was previous disabled, there is no need to manually relocate any existing database files.

```
$ bin/stop-server
$ bin/start-server
```

To Reset the LDAP Changelog Location Using dsconfig Non-Interactive Mode

1. If you have changed the LDAP Changelog location, but want to reset it to its original location, use `dsconfig` to reset it. The following command resets the LDAP changelog backend to `<server-root>/db` location. Remember to include the LDAP connection parameters (e.g., hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --reset "db-directory"
```

2. The server attempts to use whatever it finds in the configured location when it starts. If there is nothing there, it will create an empty database. If the LDAP changelog backend at the previous location is enabled at the time, stop the server, manually copy the database files to the new LDAP changelog location, and then restart the server.

Viewing the LDAP Changelog Parameters in the Root DSE

The Root DSE is a special entry that holds operational information about the server. The entry provides information about the LDAP controls, extended operations, and SASL mechanisms available in the server as well as the state of the data within the changelog. For changelog parameters, the attributes of interest include:

- **firstChangeNumber**. Change number for the first (oldest) change record contained in the LDAP changelog.
- **lastChangeNumber**. Change number for the last (most recent) change record contained in the LDAP changelog.
- **lastPurgedChangeNumber**. Change number for the last change that was purged from the LDAP changelog. It can be 0 if no changes have yet been purged.
- **firstReplicaChange**. Information about the first (oldest) change record for a change received from the specified replica. This is a multi-valued attribute and should include a value for each server in the replication topology.
- **lastReplicaChange**. Information about the last (most recent) change record for a change received from the specified replica.

The `firstReplicaChange` and `lastReplicaChange` attributes use the following syntax:

```
serverID:CSN:changeNumber
```

where:

- **serverID**. Specifies the unique identifier for the server updating the change log.
- **CSN**. Specifies the Change Sequence Number, which is the time when the update was made to the given replica.
- **changeNumber**. Specifies the order of the change that is logged to the LDAP changelog.

The `firstReplicaChange` and `lastReplicaChange` attributes can be used to correlate information in the local LDAP Change Log with data in the LDAP Change Log of other servers in the replication topology. The order of the individual changes in the LDAP Change Log can vary between servers based on the order in which they were received from a replica.

To View the LDAP Changelog Parameters

- Use `ldapsearch` to view the Root DSE.

```
$ bin/ldapsearch --baseDN "" --searchScope base "(objectclass=*)" "+"
```

Viewing the LDAP Changelog Using ldapsearch

All records in the changelog are immediate children of the `cn=changelog` entry and are named with the `changeNumber` attribute. You can view changelog entries using `ldapsearch`. Changes are represented in the form documented in the *draft-good-ldap-changelog* specification with the targetDN attribute providing the DN of the updated entry, the `changeType` attribute providing the type of operation (add, delete, modify, or modDN), and the changes attribute providing a base64-encoded representation of the attributes included in the entry (for add operations) or the changes made (for modify operations) in LDIF form. You can view the changes by decoding the encoded value using the `base64` decode utility. The LDAP SDK for Java also provides support for parsing changelog entries.

To View the LDAP Changelog Using ldapsearch

1. Grant access to the `cn=changelog` backend to the `uid=admin` account using access control rules. By default, only the root user has access to this backend.

```
$ bin/ldapmodify
dn: cn=changelog
changetype: modify
add: aci
aci: (targetattr="*||+")
  (version 3.0; acl "Access to the changelog backend for the admin
 account";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

2. Use `ldapsearch` to view the changelog.

```
$ bin/ldapsearch --baseDN cn=changelog --dontWrap "(objectclass=*)"
```

```
dn: cn=changelog
objectClass: top
objectClass: untypedObject
cn: changelog

dn: changeNumber=1,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.0,ou=People,dc=example,dc=com
changeType: modify
changes::
 cmVwbGFjZTogbW9iaWxlCm1vYmlsZTogKzEgMDIwIDE1NCA5Mzk4Ci0KcmVwbGFjZTogа
G9tZVBob25lCmhvbWVQaG9uZTogKzEgMjI1IDIxNiA0OTQ5Ci0KcmVwbGFjZTogZ2l2ZW5OYW1lCmdp
dmVuTmFtZTogQWFyb24KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogdGhpcyBpcyB
0aGUgZGVzY3JpcHRpb24gZm9yIEFhcm9uIEF0cC4KLQpyZXBsYWNlOiBtb2RpZmllcnNOYW1lCm1vZG
lmaWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VyLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwotCnJlc
GxhY2U6IGRzLXVwZGF0ZS10aW1lCmRzLXVwZGF0ZS10aW1lOjogQUFBQkhQOHpUR0E9Cgo=
changenumber: 1
dn: changeNumber=2,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: dc=example,dc=com
changeType: modify
changes::
 cmVwbGFjZTogZHMtc3luYy1zdGF0ZQpkcy1zeW5jLXN0YXRlOiAwMDAwMDExQ0ZGMzM0Q
zYwNDA5MzAwMDAwMDAyCgo=
changenumber: 2
```

To View the LDAP Change Sequence Numbers

- The changelog displays the server state information, which is important for failover between servers during synchronization operations. The server state information is exchanged between the servers in the network (LDAP servers and replication servers) as part of the protocol start message. It also helps the client application determine which server is most up-to-date. Make sure that the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

```
$ bin/ldapsearch --baseDN cn=changelog --dontWrap "(objectclass=*)" "+"
```

```
dn: cn=changelog

dn: changeNumber=1,cn=changelog
entry-size-bytes: 182
targetUniqueId: 68147342-1f61-3465-8489-3de58c532130
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184D9E303000000001
replicationCSN: 0000011D27184D9E303000000001
replicaIdentifier: 12336

dn: changeNumber=2,cn=changelog
entry-size-bytes: 263
targetUniqueId: 4e9b7847-edcb-3791-b11b-7505f4a55af4
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184F2E303000000002
replicationCSN: 0000011D27184F2E303000000002
replicaIdentifier: 12336
```

To View LDAP Changelog Monitoring Information

The changelog contains a monitor entry that you can access over LDAP, JConsole, the Administrative Console, or SNMP. Make sure that the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

- Use `ldapsearch` to view the changelog monitor entry.

```
$ bin/ldapsearch --baseDN cn=changelog,cn=monitor "(objectclass=*)"
```

```
dn: cn=changelog,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: extensibleObject
cn: changelog
changelog: cn=changelog
firstchangenumber: 1
lastchangenumber: 8
lastpurgedchangenumber: 0
firstReplicaChange: 16225:0000011D0205237F3F6100000001:5
firstReplicaChange: 16531:0000011CFF334C60409300000002:1
lastReplicaChange: 16225:0000011D02054E8B3F6100000002:7
lastReplicaChange: 16531:0000011CFF334C60409300000002:1
oldest-change-time: 20081015063104Z
...(more data)...
```

Indexing the LDAP Changelog

The Directory Server supports attribute indexing in the Changelog Backend to allow Get Changelog Batch requests to filter results that include only changes involving specific attributes. Normally, the directory server that receives a request must iterate over the whole range of changelog entries and then match entries based on search criteria for inclusion in the batch. The majority of this processing also involves determining whether the changelog entry includes changes to a particular attribute or set of attributes, or not. Using changelog indexing, client applications can dramatically speed up throughput when targeting the specific attributes.

Administrators can configure attribute indexing using the `index-include-attribute` and `index-exclude-attribute` properties on the Changelog Backend. The properties can accept the specific attribute name or special LDAP values "*" to specify all user attributes or "+" to specify all operational attributes.

To determine if the directory server supports this feature, administrators can view the Root DSE for the following entry:

```
supportedFeatures: 1.3.6.1.4.1.30221.2.12.3
```

To Index a Changelog Attribute

1. Use `dsconfig` to set attribute indexing on an attribute in the Changelog Backend.

   The following command enables the Changelog Backend and sets the backend to include all user attributes ("*") for ADD or MODIFY operations using the `changelog-include-attribute` property. The `changelog-deleted-entry-include-attribute` property is set to all attributes ("*") to specify a set of attribute types that should be included in a changelog entry for DELETE operations. Attributes specified in this list will be recorded in the `deletedEntryAttrs` attribute on the changelog entry when an entry is deleted. The attributes `displayName` and `employeeNumber` are indexed using the `index-include-attribute` property.

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --set "enabled:true" \
     --set "changelog-include-attribute:*" \
     --set "changelog-deleted-entry-include-attribute:*" \
     --set "index-include-attribute:displayName" \
     --set "index-include-attribute:employeeNumber"
   ```
2. Add another attribute to index using the `dsconfig` --add option, which adds the attribute to an existing configuration setting.

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --add "index-include-attribute:cn"
   ```

To Exclude Attributes from Indexing

- Use `dsconfig` to set attribute indexing on all user attributes in the Changelog Backend. The following command includes all user attributes except the description and location attributes.

  ```
  $ bin/dsconfig set-backend-prop --backend-name changelog \
    --set "index-include-attribute:*" \
    --set "index-exclude-attribute:description \
    --set "index-exclude-attribute:location
  ```

Tracking Virtual Attribute Changes in the LDAP Changelog

By default, the LDAP Changelog tracks changes to real attributes only. For client applications that require change tracking to include virtual attributes, administrators can enable the `include-virtual-attribute` property, so that real and virtual attributes are tracked within the changelog. Once the `include-virtual-attribute` property is enabled, then properties for virtual attributes that store before/after values, key attributes, and added or deleted entry attributes can be enabled.

To Track Virtual Attribute Changes in the LDAP Changelog

- Use `dsconfig` to enable virtual attribute change tracking in the LDAP Changelog.

  The following command enables the LDAP changelog and sets `include-virtual-attributes` to `add-attributes`, which indicates that virtual attribute be included in the set of attributes listed for an add operation. The `delete-entry-attributes` option indicates that virtual attributes should be included in the set of deleted entry attributes listed for a delete operation. The `before-and-after-values` option indicates that virtual attributes should be included in the set of before and after values for attributes targeted by the changes.

The `key-attribute-values` option indicates that virtual attributes should be included in the set of entry key attribute values.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "enabled:true" \
  --set "include-virtual-attributes:add-attributes" \
  --set "include-virtual-attributes:deleted-entry-attributes" \
  --set "include-virtual-attributes: before-and-after-values" \
  --set "include-virtual-attributes: key-attribute-values"
```

Chapter 16

Managing Access Control

Topics:

The PingDirectory Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and provides examples that illustrate the use of key access control functionality.

Overview of Access Control

The access control model uses access control instructions (ACIs), which are stored in the `aci` operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and affects the entry or any subentries within that branch of the directory information tree (DIT).

Access control instructions specifies four items:

- **Resources**. *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the `cn` or `userPassword` password.
- **Name**. *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."
- **Clients**. *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: `groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."`
- **Rights**. *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant `read` or `write` permission to a `telephoneNumber` attribute.

Key Access Control Features

The PingDirectory Server provides important access control features that provide added security for the Directory Server's entries.

Improved Validation and Security

The Directory Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed `aci` value will be rejected.

As an additional level of security, the Directory Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the `lockdown-mode` privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to an access control instruction not being enforced. When the problem has been corrected, the administrator can use the `leave-lockdown-mode` tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the server (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the PingDirectory Server, global ACIs are defined within the server configuration, in the `global-aci` property of configuration object for the access control handler. They can be viewed and managed using configuration tools like `dsconfig` and the Administrative Console.

The global ACIs available by default in the PingDirectory Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: `namingContexts`, `subschemaSubentry`, `supportedAuthPasswordSchemes`, `supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedLDAPVersion`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`.

- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

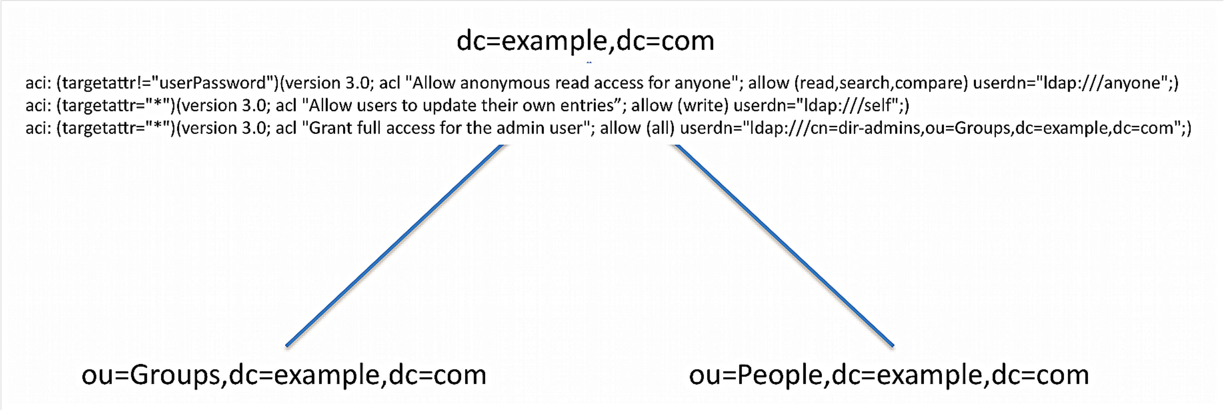
Access Controls for Public or Private Backends

The PingDirectory Server classifies backends as either public or private, depending on their intended purpose. A private backend is one whose content is generated by the Directory Server itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by the server (for example, the LDAP changelog backend). A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The PingDirectory Server access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General Format of the Access Control Rules

Access control instructions (ACIs) are represented as strings that are applied to one or more entries within the Directory Information Tree (DIT). Typically, an ACI is placed on a subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Directory Server iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissable operations. When a client application, such as `ldapsearch`, enters a request, the Directory Server checks that the user who binds with the server has the necessary access rights to the requested search targets. ACIs are cumulatively applied, so that a user who may have an ACI at an entry, may also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.



```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous read access for anyone"; allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")(version 3.0; acl "Allow users to update their own entries"; allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin user"; allow (all) userdn="ldap:///cn=dir-admins,ou=Groups,dc=example,dc=com";)
```

Figure 5: ACI

An access control rule has a basic syntax as follows:

```
aci : (targets) (version 3.0; acl "name"; permissions bind rules;)
```

Table 28: Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and/or attributes to which an access control rule applies. Syntax: *(target keyword = || != expression)*
name	Specifies the name of the ACI.
permissions	Specifies the type of operations to which an access control rule might apply. Syntax: *allow||deny (permission)*
bind rules	Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: *bind rule keyword = ||!= expression;*. The bind rule syntax requires that it be terminated with a ";".

Summary of Access Control Keywords

This section provides an overview of the keywords supported for use in the PingDirectory Server access control implementation.

Targets

A target expression specifies the set of entries and/or attributes to which an access control rule applies. The *keyword* specifies the type of target element. The *expression* specifies the items that is targeted by the access control rule. The operator is either the equal ("=") or not-equal ("!="). Note that the "!=" operator cannot be used with `targattrfilters` and `targetscope` keywords. For specific examples on each target keyword, see the section *Working with Targets*.

```
(keyword [=||!=]expression)
```

The following keywords are supported for use in the target portion of ACIs:

Table 29: Summary of Access Control Target Keywords

Target Keyword	Description	Wildcards
extop	Specifies the OIDs for any extended operations to which the access control rule should apply.	No
target	Specifies the set of entries, identified using LDAP URLs, to which the access control rule applies.	**Yes**
targattrfilters	Identifies specific attribute values based on filters that may be added to or removed from entries to which the access control rule applies.	**Yes**
targetattr	Specifies the set of attributes to which the access control rule should apply.	**Yes**
targetcontrol	Specifies the OIDs for any request controls to which the access control rule should apply.	No
targetfilter	Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.	**Yes**
targetscope	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI fi there is no target, to which the access control rule should apply.	No

Permissions

Permissions indicate the types of operations to which an access control rule might apply. You can specify if the user or group of users are allowed or not allowed to carry out a specific operation. For example, you would grant read access to a targeted entry or entries using "allow (read)" permission. Or you can specifically deny access to the target

entries and/or attributes using the "deny (read)" permission. You can list out multiple permissions as required in the ACI.

```
allow (permission1 ...,permission2,...permissionN)
```

```
deny (permission1 ...,permission2,...permissionN)
```

The following keywords are supported for use in the permissions portion of ACIs:

Table 30: Summary of Access Control Permissions

Permission	Description
add	Indicates that the access control should apply to add operations.
compare	Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
delete	Indicates that the access control should apply to delete operations.
export	Indicates that the access control should only apply to modify DN operations in which an entry is moved below a different parent by specifying a new superior DN in the modify DN request. The requestor must have the `export` permission for operations against the entry's original DN. The requestor must have the `import` permission for operations against the entry's new superior DN. For modify DN operations that merely alter the RDN of an entry but keeps it below the same parent (i.e., renames the entry), only the `write` permission is required. This is true regardless of whether the entry being renamed is a leaf entry or has subordinate entries.
import	See the description for the `export` permission.
proxy	Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identify to be specified).
read	Indicates that the access control rule should apply to search result entries returned by the server.
search	Indicates that the access control rule should apply to search operations with a non-base scope.
selfwrite	Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
write	Indicates that the access control rule should apply to modify and modify DN operations.
all	An aggregate permission that includes all other permissions except “proxy.” This is equivalent to providing a permission of “add, compare, delete, read, search, selfwrite, write, export, and import.”

Bind Rules

The Bind Rules indicate whether an access control rule should apply to a given requester. The syntax for the target keyword is shown below. The *keyword* specifies the type of target element. The expression specifies the items that is targeted by the access control rule. The operator is either equals ("=") or not-equals ("!="). The semi-colon delimiter symbol (";") is required after the end of the final bind rule.

```
keyword [=||!= ] expression;
```

Multiple bind rules can be combined using boolean operations (AND, OR, NOT) for more access control precision. The standard Boolean rules for evaluation apply: innermost to outer parentheses first, left to right expressions, NOT before AND or OR. For example, an ACI that includes the following bind rule targets all users who are not `uid=admin,dc=example,dc=com` and use simple authentication.

```
(userdn!="ldap:///uid=admin,dc=example,dc=com" and authmethod="simple");
```

The following bind rule targets the `uid=admin,dc=example,dc=com` and authenticates using SASL EXTERNAL or accesses the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or
 ip="127.0.0.1"));
```

The following keywords are supported for use in the bind rule portion of ACIs:

Table 31: Summary of Bind Rule Keywords

Bind Rule Keyword	Description

authmethod	Indicates that the requester’s authentication method should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword’s syntax is as follows:
	`authmethod = method`
	where *method* is one of the following representations:
	none
	simple. Indicates that the client is authenticated to the server using a bind DN and password.
	ssl. Indicates that the client is authenticated with an SSL/TLS certificate (e.g., via SASL EXTERNAL), and not just over a secure connection to the server.
	sasl {sasl_mechanism}. Indicates that the client is authenticated to the server using a specified SASL Mechanism.
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (userdn="ldap:///self" and authmethod="ssl");)`

dayofweek	Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. Multiple day of week values may be separated by commas. The keyword’s syntax is as follows:
	`dayofweek = day1, day2, ...`
	where *day* is one of the following representations:
	sun
	mon
	tues
	wed
	thu
	fri
	sat
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) on weekdays to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`

<table>
<tr><th>Bind Rule Keyword</th><th>Description</th></tr>
<tr><td></td><td><code>   allow (write) (dayofweek!="sun,sat" and userdn="ldap:///</code>
<code>self"</code>
<code>   and authmethod="ssl");)</code></td></tr>
<tr><td>dns</td><td>Indicates that the requester’s DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple DNS patterns may be separated by commas. The keyword’s syntax is as follows:
<code><b>dns</b> = <i>dns-host-name</i></code>
The following example allows users on hostname <code>server.example.com</code> to update their own entries:
<code>aci: (targetattr="*")</code>
<code> (version 3.0; acl "Allow users to update their own entries";</code>
<code>   allow (write) (dns="server.example.com" and userdn="ldap:///</code>
<code>self");)</code></td></tr>
<tr><td>groupdn</td><td>Indicates that the requester’s group membership should be taken into account when determining whether the access control rule should apply to any operation. Wildcards are not allowed in this expression.
<code><b>groupdn [= || !=] "ldap:///groupdn [|| ldap:///</b></code>
<code><b>groupdn] ..."</b></code>
The following example allows users in the managers group to update their own entries:
<code>aci: (targetattr="*")</code>
<code> (version 3.0; acl "Allow users to update their own entries";</code>
<code>   allow (write)</code>
<code>   (groupdn="ldap:///</code>
<code>cn=managers,ou=groups,dc=example,dc=com");)</code></td></tr>
<tr><td>ip</td><td>Indicates that the requester’s IP address should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple IP address patterns may be separated by commas. The keyword’s syntax is as follows:
<code><b>ip</b> [= || !=] <i>ipAddressList</i></code>
where <i>ipAddressList</i> is one of the following representations:
A specific IPv4 address: 127.0.0.1
An IPv4 address with wildcards to specify a subnetwork: 127.0.0.*
An IPv4 address or subnetwork with subnetwork mask: 123.4.5.0+255.255.255.0
An IPv4 address range using CIDR notation: 123.4.5.0/24
An IPv6 address as defined by RFC 2373.
The following example allows users on 10.130.10.2 and localhost to update their own entries:
<code>aci: (targetattr="*")</code>
<code> (version 3.0; acl "Allow users to update their own entries";</code>
<code>   allow (write) (ip="10.130.10.2,127.0.0.1" and</code>
<code> userdn="ldap:///self");)</code></td></tr>
</table>

Bind Rule Keyword	Description

timeofday	Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	`timeofday [ = || != || >= || > || <= || < ] time`
	where *time* is one of the following representations:
	4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour)
	Wildcards are not allowed in this expression
	The following example allows users to update their own entries if the request is received before 12 noon.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users who authenticate before noon`
	`    to update their own entries";`
	`    allow (write) (timeofday<1200 and userdn="ldap:///self"`
	`    and authmethod="simple");)`

userattr	Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of `USERDN` indicates that the target attribute should have a value which matches the DN of the authenticated user. A bindType value of `GROUPDN` indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A bindType value of `LDAPURL` indicates that the target attribute should have a value that is an LDAP URL whose criteria matches the entry for the authenticated user. Any value other than `USERDN`, `GROUPDN`, or `LDAPURL` is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:
	`userattr = attrName# [ bindType || attrValue ]`
	where:
	attrName = name of the attribute for matching
	bindType = USERDN, GROUPDN, LDAPURL
	attrValue = an attribute value. Note that the *attrVALUE* of the attribute must match on both the bind entry and the target of the ACI.
	The following example allows a manager to change employee's entries. If the bind DN is specified in the *manager* attribute of the targeted entry, the bind rule is evaluated to TRUE.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a manager to change employee`
	` entries";`
	`   allow (write) userattr="manager#USERDN";)`
	The following example allows any member of a group to change employee's entries. If the bind DN is a member of the group specified in the *allowEditors* attribute of the targeted entry, the bind rule is evaluated to TRUE.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow allowEditors to change employee`
	` entries";`
	`   allow (write) userattr="allowEditors#GROUPDN";)`

Bind Rule Keyword	**Description**

	The following example allows allows a user's manager to edit that user's entry and any entries below the user's entry up to two levels deep. You can specify up to five levels (0, 1, 2, 3, 4) below the targeted entry, with zero (0) indicating the targeted entry.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow managers to change employees entries`
	` two levels below";`
	`    allow (write) userattr="parent[0,1,2].manager#USERDN";)`
	The following example allows any member of the engineering department to update any other member of the engineering department at or below the specified ACI.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow any member of Eng Dept to update any`
	` other member of the`
	`   enginering department at or below the ACI";`
	`   allow (write) userattr="department#ENGINEERING";)`
	The following example allows an entry to be updated by any user whose entry matches the criteria defined in the LDAP URL contained in the `allowedEditorCriteria` attribute of the target entry.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a user that matches the filter to`
	` change entries";`
	`   allow (write) userattr="allowedEditorCriteria#LDAPURL";)`

userdn	Indicates that the user's DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:
	`userdn [ = || != ] "ldap:///value [ || "ldap:///value ..."]`
	where *value* is one of the following representations:
	The DN of the target user
	A value of `anyone` to match any client, including unauthenticated clients.
	A value of `all` to match any authenticated client.
	A value of `parent` to match the client authenticated as the user defined in the immediate parent of the target entry.
	A value of `self` to match the client authenticated as the user defined in the target entry.
	If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.
	The following example allows users to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`    allow (write) userdn="ldap:///self";)`

Working with Targets

The following section presents a detailed look and examples of the target ACI keywords: `target`, `targetattr`, `targetfilter`, `targattrfilters`, `targetscope`, `targetcontrol`, and `extop`.

target

The `target` keyword indicates that the ACI should apply to one or more entries at or below the specified distinguished name (DN). The target DN must be equal or subordinate to the DN of the entry in which the ACI is placed. For example, if you place the ACI at the root of `ou=People,dc=example,dc=com`, you can target the DN, `uid=user.1,ou=People,dc=example,dc=com` within your ACI rule. The DN must meet the string representation specification of distinguished names, outlined in RFC 4514, and requires that special characters be properly escaped.

The `target` clause has the following format, where DN is the distinguished name of the entry or branch:

```
(target = ldap:///DN)
```

For example, to target a specific entry, you would use a clause such as the following:

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

Note that, in general, specifying a target DN is not recommended. It is better to have the ACI defined in that entry and omit the `target` element altogether. For example, although you can have `(target="ldap:///uid=john.doe,ou=People,dc=example,dc=com)` in any of the `dc=example,dc=com` or `ou=People` entries, it is better for it to be defined in the `uid=john.doe` entry and not explicitly include the `target` element.

The expression allows for the "not equal" (!=) operator to indicate that all entries within the scope of the given branch that do NOT match the expression be targeted for the ACI. Thus, the following expression targets all entries within the subtree that do not match `uid=john.doe`.

```
(target != ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

The `target` keyword also supports the use of asterisk (*) characters as wildcards to match elements within the distinguished name. The following target expression matches all entries that contains and begins with "john.d, " so that entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=People,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
```

The following target expression matches all entries whose DN begins with "john.d," and matches the `ou` attribute. Entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=asia-branch,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
```

Another example of a complete ACI targets the entries in the `ou=People,dc=example,dc=com` branch and the entries below it, and grants the users the privilege to modify all of their user attributes within their own entries.

```
aci:(target="ldap:///ou=People,dc=example,dc=com")
  (targetattr="*")
  (version 3.0; acl "Allow all the ou=People branch to modify their own
 entries";
 allow (write) userdn="ldap:///self";)
```

targetattr

The `targetattr` keyword targets the attributes for which the access control instruction should apply. There are four general forms that it can take in the PingDirectory Server:

- **(targetattr="*")**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **(targetattr="+")**. Indicates that the access control rule applies to all operational attributes. User attributes will not automatically be included in this set.
- **(targetattr="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies only to the named set of attributes.

- **(targetattr!="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as user attributes and operational attributes. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes, such as when the entry was created, last modified and by whom. Metadata can also include attributes specifying which password policy applies to the user, or overrided default constraints like size limit, time limit, or look-through limit for that user.

The PingDirectory Server distinguishes between these two types of attributes in its access control implementation. The Directory Server does not automatically grant any access at all to operational attributes. For example, the following clause applies only to user attributes and not to operational attributes:

```
(targetattr="*")
```

You can also target multiple attributes in the entry. The following clause targets the common name (cn), surname (sn) and state (st) attribute:

```
(targetattr="cn||sn||st")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

To include all user and all operational attributes, you use both symbols, as follows:

```
(targetattr="*||+")
```

If there is a need to target a specific operational attribute rather than all operational attributes, then it can be specifically included in the values of the `targetattr` clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or if you want to target all user attributes and a specific operational attribute, then you can use them in the `targetattr` clause, as follows:

```
(targetattr="*||ds-rlim-size-limit")
```

The following ACIs are placed on the `dc=example,dc=com` tree and allows any user anonymous read access to all entries except the `userPassword` attribute. The second ACI allows users to update their own contact information. The third ACI allows the `uid=admin` user full access privileges to all user attributes in the `dc=example,dc=com` subtree.

```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous
   read access for anyone"; allow (read,search,compare) userdn="ldap:///
anyone";)
aci: (targetattr="telephonenumber||street||homePhone||l||st")
  (version 3.0; acl "Allow users to update their own contact info";
   allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin
 user";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

An important note must be made when assigning access to user and operational attributes, which can be outlined in an example to show the implications of the Directory Server not distinguishing between these attributes. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of `uid` and `employeeNumber`. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a Directory Server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users including administrators granted full rights by access control rules, but excluding users with the `bypass-acl` privilege, allow them to edit any other attributes in their own entry including those excluded by rules like `uid` and `employeeNumber` in the example above, or add, modify, or delete any entries below his or her own entry.

Because the PingDirectory Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the `modify-acl` privilege.

Another danger in using the `(targetattr!="x")` pattern is that two ACIs within the same scope could have two different `targetattr` policies that cancel each other out. For example, if one ACI has `(targetattr!="cn||sn")` and a second ACI has `(targetattr!="userPassword")`, then the net effect is `(targetattr="*")`, because the first ACI inherently allows `userPassword`, and the second allows `cn` and `sn`.

targetfilter

The `targetfilter` keyword targets all attributes that match results returned from a filter. The `targetfilter` clause has the following syntax:

```
(targetfilter = ldap_filter)
```

For example, the following clause targets all entries that contain "ou=engineering" attribue:

```
(targetfilter = "(ou=engineering)")
```

You can only specify a single filter, but that filter can contain multiple elements combined with the OR operator. The following clause targets all entries that contain "ou=engineering," "ou=accounting," and "ou=marketing."

```
(targetfilter = "(|(ou=engineering)(ou=accounting)(ou=marketing)")
```

The following example allows the user, `uid=eng-mgr`, to modify the `departmentNumber`, `cn`, and `sn` attributes for all entries that match the filter `ou=engineering`.

```
aci:(targetfilter="(ou=engineering)")
  (targetattr="departmentNumber||cn||sn")
  (version 3.0; acl "example"; allow (write)
   userdn="ldap:///uid=eng-mgr,dc=example,dc=com";)
```

targattrfilters

The `targattrfilters` keyword targets specific attribute *values* that match a filtered search criteria. This keyword allows you to set up an ACI that grants or denies permissions on an attribute value if that value meets the filter criteria. The `targattrfilters` keyword applies to individual values of an attribute, not to the whole attribute. The keyword also allows the use of wildcards in the filters.

The keyword clause has the following formats:

```
(target = "add=attr1:Filter1 && attr2:Filter2... && attrn:FilterN,
del=attr1:Filter1 && attr2:Filter2 ... && attrN:FilterN" )
```

where

add represents the operation of adding an attribute value to the entry
del represents the operation of removing an attribute value from the entry
attr1, attr2... attrN represents the targeted attributes
filter1, filter2 ... filterN represents filters that identify matching attribute values

The following conditions determine when the attribute must satisfy the filter:

- When adding or deleting an entry containing an attribute targeted a `targattrfilters` element, each value of that attribute must satisfy the corresponding filter.
- When modifying an entry, if the operation adds one or more values for an attribute targeted by a `targattrfilters` element, each value must satisfy the corresponding filter. If the operation deletes one or more values for a targeted attribute, each value must satisfy the corresponding filter.
- When replacing the set of values for an attribute targeted by a `targattrfilters` element, each value removed must satisfy the delete filters, and each value added must satisfy the add filters.

The following example allows any user who is part of the `cn=directory server admins` group to add the `soft-delete-read` privilege.

```
aci:(targattrfilter="add=ds-privilege-name:(ds-privilege-name=soft-delete-
read)")
  (version 3.0; acl "Allow members of the directory server admins group to
 grant the
   soft-delete-read privilege"; allow (write)
   groupdn="ldap:///cn=directory server admins,ou=group,dc=example,dc=com";)
```

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is base to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
      (targetattr="supportedControl||supportedExtension||
       supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
      (version 3.0; acl "Allow users to view Root DSE Operational Attributes";
       allow (read,search,compare) userdn="ldap:///anyone")
```

targetcontrol

The `targetcontrol` keyword is used to indicate whether a given request control can be used by those users targeted in the ACI. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying control OIDs.

The following ACI example shows the controls required to allow an administrator to use and manage the Soft-Delete feature. The Soft Delete Request Control allows the user to soft-delete an entry, so that it could be undeleted at a later time. The Hard Delete Request Control allows the user to permanently remove an entry or soft-deleted entry. The Undelete Request Control allows the user to undelete a currently soft-deleted entry. The Soft-Deleted Entry Access Request Control allows the user to search for any soft-deleted entries in the server.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.22||
   1.3.6.1.4.1.30221.2.5.23||1.3.6.1.4.1.30221.2.5.24")
  (version 3.0; acl "Allow admins to use the Soft Delete Request Control,
   Hard Delete Request Control,Undelete Request Control, and
   Soft-deleted entry access request control";
   allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

extOp

The `extop` keyword can be used to indicate whether a given extended request operation can be used. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying extended request OIDs.

The following ACI allows the `uid=user-mgr` to use the Password Modify Request (i.e., OID=1.3.6.1.4.1.4203.1.11.1) and the StartTLS (i.e., OID=1.3.6.1.4.1.1466.20037) extended request OIDs.

```
aci:(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")
  (version 3.0; acl "Allows the mgr to use the Password Modify Request and
 StartTLS;
   allow(read) userdn="ldap:///uid=user-mgr,ou=people,dc=example,dc=com";)
```

Examples of Common Access Control Rules

This section provides a set of examples that demonstrate access controls that are commonly used in your environment. Note that to be able to alter access control definitions in the server, a user must have the `modify-acl` privilege as discussed later in this chapter.

Administrator Access

The following ACI can be used to grant any member of the "`cn=admins,ou=groups,dc=example,dc=com`" group to add, modify and delete entries, reset passwords and read operational attributes such as `isMemberOf` and password policy state:

```
aci: (targetattr="+")(version 3.0; acl "Administrators can read, search or
 compare operational attributes";
allow (read,search,compare) groupdn="ldap:///
cn=admins,ou=groups,dc=example,dc=com";)
aci: (targetattr="*")(version 3.0; acl "Administrators can add, modify and
 delete entries";
allow (all) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

Anonymous and Authenticated Access

The following ACI allow anonymous read, search and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. The authenticated user will inherit the privileges of the anonymous ACI. In addition, the authenticated user can change `userPassword`:

```
aci: (targetattr="objectclass || uid || cn || mail || sn || givenName")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Anyone can access names and email addresses of entries
 representing people";
allow (read,search,compare) userdn="ldap:///anyone";)
```

```
aci: (targetattr="departmentNumber || manager || isMemberOf")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Authenticated users can access these fields for entries
 representing people";
allow (read,search,compare) userdn="ldap:///all";)
aci: (targetattr="userPassword")(version 3.0; acl "Authenticated users can
 change password";
allow (write) userdn="ldap:///all";)
```

If no unauthenticated access should be allowed to the Directory Server, the preferred method for preventing unauthenticated, or anonymous access is to set the Global Configuration property `reject-unauthenticated-requests` to true.

Delegated Access to a Manager

The following ACI can be used to allow an employee's manager to edit the value of the employee's `telephoneNumber` attribute. This ACI uses the `userattr` keyword with a bind type of `USERDN`, which indicates that the target entry's manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
(version 3.0; acl "A manager can update telephone numbers of her direct
 reports";
allow (read,search,compare,write) userattr="manager#USERDN";)
```

Proxy Authorization

The following ACIs can be used to allow the application "`cn=OnBehalf,ou=applications,dc=example,dc=com`" to use the proxied authorization v2 control to request that operations be performed using an alternate authorization identity. The application user is also required to have the `proxied-auth` privilege as discussed later in this chapter:

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com";)
```

Validating ACIs Before Migrating Data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Sun/Oracle encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the PingDirectory Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a PingDirectory Server.

To validate an access control instruction, the PingDirectory Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values, and all other attributes will be ignored.

To Validate ACIs from a File

The `validate-acis` tool can process data contained in an LDIF file. It will ignore all attributes except `aci`, and will ignore all entries that do not contain the `aci` attribute, so any existing LDIF file that contains access control rules may be used.

1. Run the `bin/validate-acis` tool (UNIX or Linux systems) or `bat\validate-acis` (Win dows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

   ```
   $ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-
   acis.ldif
   ```

   ```
   # Processing complete # Total entries examined: 1
   # Entries found with ACIs: 1
   # Total ACI values found: 3
   # Malformed ACI values found: 0
   # Other processing errors encountered: 0
   ```
2. Review the results by opening the output file. For example, the `validated-acis.ldif` file that was generated in the previous step reads as follows:

   ```
   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr!="userPassword")
     (version 3.0; acl "Allow anonymous read access for anyone";
       allow (read,search,compare) userdn="ldap:///anyone";)

   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr="*")
     (version 3.0; acl "Allow users to update their own entries";
       allow (write) userdn="ldap:///self";)

   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr="*")
     (version 3.0; acl "Grant full access for the admin user";
       allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
   ```
3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed ACI syntax is encountered.

   ```
   $ bin/validate-acis --ldifFIle test-acis.ldif --outputFile validated-
   acis.ldif \
     --onlyReportErrors
   ```

   ```
   # Processing complete
   # Total entries examined: 1
   # Entries found with ACIs: 1
   # Total ACI values found: 3
   # Malformed ACI values found: 0
   # Other processing errors encountered: 0
   ```

   The output file shows the following message:

   ```
   # The following access control rule is malformed or contains an unsupported
   # syntax: The provided string '(targetattr!=userPassword)(version 3.0; acl
   # "Allow anonymous read access for anyone"; allow (read,search,compare)
   # userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
   # Instruction (ACI) because it failed general ACI syntax evaluation
   dn: dc=example,dc=com
   aci: (targetattr!=userPassword)
     (version 3.0; acl "Allow anonymous read access for anyone";
       allow (read,search,compare) userdn="ldap:///anyone";)

   # The following access control rule is valid
   ```

```
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Validate ACIs in Another Directory Server

The `validate-acis` tool also provides the ability to examine ACIs in data that exists in another Directory Server that you are planning to migrate to the PingDirectory Server. The tool helps to determine whether the Ping Identity Server accepts those ACIs.

- To use it in this manner, provide arguments that specify the address and port of the target Directory Server, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

Migrating ACIs from Sun/Oracle to PingDirectory Server

This section describes the most important differences in access control evaluation between Sun/Oracle and the PingDirectory Server.

Support for Macro ACIs

Sun/Oracle provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the PingDirectory Server. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN Bind Rule

Sun/Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The PingDirectory Server does not support DSEE roles and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting Operational Attributes

The Sun/Oracle access control model does not differentiate between user attributes and operational attributes. With Sun/Oracle, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible for several significant security holes in which users are unintentionally given access to operational attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the PingDirectory Server, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except

`userPassword`, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like `targetattr="creatorsName||modifiersName"`. All operational attributes can be targeted using the `"+"` character. So, `targetattr="+"` targets all operational attributes but no user attributes and `targetattr="*||+"` targets all user and operational attributes.

Specification of Global ACIs

Both DSEE and PingDirectory Server support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the `aci` attribute. In the PingDirectory Server, global ACIs are managed with `dsconfig` referenced in the `global-aci` property of the Access Control Handler.

Defining ACIs for Non-User Content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the PingDirectory Server, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting Access to Controls and Extended Operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as "`oid={oid},cn=features,cn=config`" where `{oid}` is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control: "`oid=2.16.840.1.113730.3.4.3,cn=features,cn=config`".

In the PingDirectory Server, the "`targetcontrol`" keyword can be used to define ACIs that grant or deny access to controls. The "`extop`" keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for Malformed ACI Values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The PingDirectory Server is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the `lockdown-mode` privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.

Note: Consider running the `import-ldif` tool with the --rejectFile option so that you can review any rejected ACIs.

About the Privilege Subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you cannot restrict the capabilities assigned to that root user.

The PingDirectory Server offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server

configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying Unsupported ACIs

The PingDirectory Server provides a `validate-acis` tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the PingDirectory Server instance. When migrating data from a DSEE deployment into a PingDirectory Server instance, the `validate-acis` tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the PingDirectory Server.

For more information about using this tool, see *Validating ACIs Before Migrating Data*.

Working with Privileges

In addition to the access control implementation, the PingDirectory Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the PingDirectory Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available Privileges

The following privileges are defined in the PingDirectory Server.

Table 32: Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the `audit-data-security` tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the `bypass-acl` privilege but without the `config-read` privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is not evaluated for bind

Privilege	Description
	operations so that password policy evaluation will still occur when binding as a user with this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the `config-read` privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the Directory Server. The user is also required to have the `jmx-read` privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Server via the Java Management Extensions (JMX). The user is also required to have the `jmx-read` privilege. Note that currently all of the information exposed by the server over JMX is read-only.
ldif-export	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group.
ldif-import	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the `ldif-export` privilege.
lockdown-mode	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.
modify-acl	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.
password-reset	This privilege is required for one user to be allowed to change another user's password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.
privilege-change	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the `ds-privilege-name` operational attribute.

Privilege	Description
proxied-auth	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
server-restart	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
server-shutdown	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
soft-delete-read	This privilege is required for a user to access a soft-deleted-entry.
stream-values	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
unindexed-search	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
update-schema	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges Automatically Granted to Root Users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the `ds-cfg-inherit-default-root-privileges` attribute with a value of FALSE in the entries for those root users.
- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the `default-root-privilege-name` property of the Root DN configuration object. By default, this set of privileges includes:

audit-data-security
backend-backup
backend-restore
bypass-acl
config-read
config-write
disconnect-client
ldif-export
lockdown-mode
manage-topology
metrics-read
modify-acl
password-reset

permit-get-password-policy-state-issues
privilege-change
server-restart
server-shutdown
soft-delete-read
stream-values
unindexed-search
update-schema

The privileges not granted to root users by default includes:

bypass-pw-policy
bypass-read-acl
jmx-read
jmx-write
jmx-notify
permit-externally-processed-authentication
permit-proxied-mschapv2-details
proxied-auth

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the `config-read`, `config-write`, and `privilege-change` privileges, as well as either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning Additional Privileges for Administrators

To allow access to the Tasks backend, set up a global ACI that allows access to members of an Administrators group as follows:

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*||+")
        (version 5.0; acl "Access to the tasks backend for administrators";
         allow (all) groupdn="ldap:///
         cn=admins,ou=groups,dc=example,dc=com";)'
```

Assigning Privileges to Normal Users and Individual Root Users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling Privileges

Although the privilege subsystem in the PingDirectory Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Working with Proxied Authorization

The Directory Server supports the Proxied Authorization Control (RFC 4370) to allow an authorized LDAP client to authenticate to the server as another user. Typically, LDAP servers are deployed as backend authentication systems that store user credentials and authorization privileges necessary to carry out an operation. Single sign-on (SSO) systems can retrieve user credentials from the Directory Server and then issue permissions that allow the LDAP client to request operations under the identity as another user. The use of the proxied authorization control provides a means for client applications to securely process requests without the need to bind or re-authenticate to the server for each and every operation.

The Directory Server supports the proxied authorization V1 and V2 request controls. The proxied authorization V1 request control is based on early versions of the draft-weltman-ldapv3-proxy Internet draft and is available primarily for legacy systems. It is recommended that deployments use the proxied authorization V2 request control based on RFC 4370.

The proxied authorization V2 control is used to request that the associated operation be performed as if it has been requested by some other user. This control may be used in conjunction with add, delete, compare, extended, modify, modify DN, and search requests. In that case, the associated operation will be processed under the authority of the specified authorization identity rather than the identity associated with the client connection (i.e., the user as whom that connection is bound). The target authorization identity for this control is specified as an "authzid" value, which should be either "dn:" followed by the distinguished name of the target user, or "u:" followed by the username.

Note that because of the inherent security risks associated with the use of the proxied authorization control, most directory servers that support its use enforce strict restrictions on the users that are allowed to request this control. If a user attempts to use the proxied authorization V2 request control and does not have sufficient permission to do so, then the server will return a failure response with the `AUTHORIZATION_DENIED` result code.

Configuring Proxied Authorization

Configuring proxied authorization requires a combination of access control instructions and the `proxied-auth` privilege to the entry that will perform operations as another user.

Note: You cannot use the `cn=Directory Manager` root DN as a proxying DN.

To Configure Proxied Authorization

1. Open a text editor and create a user entry, such as `uid=clientApp`, which is the user entry that will request operations as another user, `uid=admin,dc=example,dc=com`. The client application entry also requires the `proxied-auth` privilege to allow it to run proxied authorization requests. Save the file as `add-user.ldif`.

```
dn: ou=Applications,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
objectClass: extensibleObject
ou: Admins
ou: Applications

dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
givenName: Client
uid: clientApp
cn: Client App
sn: App
userPassword: password
ds-privilege-name: proxied-auth
```

2. Add the file using `ldapmodify`.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

3. The client application targets a specific subtree in the Directory Information Tree (DIT) for its operations. For example, some client may need access to an accounts subtree to retrieve customer information. Another client may need access to another subtree, such as a subscriber subtree. In this example, we want the client application to target the `ou=People,dc=example,dc=com` subtree. To allow the target, open a text editor and create an LDIF file to assign an ACI to that branch so that the client app user can access it as a proxy auth user. Note that the ACI should be on a single line of text. The example shows the ACI over multiple lines for readability. Add the file using the `ldapmodify`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci:  (version 3.0; acl "People Proxy Access"; allow(proxy)
  userdn="ldap:///uid=clientApp,ou=Applications,dc=example,dc=com";)
```

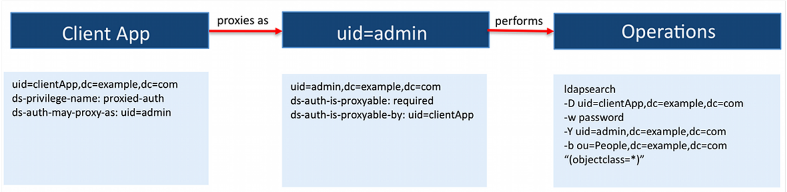
4. Run a search to test the configuration using the bind DN `uid=clientApp` and the `proxyAs` option, which requires that you prefix "dn:" to the proxying entry or "u:" to the username. The `uid=clientApp` binds to the server and proxies as `uid=admin` to access the `ou=People,dc=example,dc=com` subtree.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

Restricting Proxy Users

The Directory Server provides a set of operational attributes that restricts the proxied authorization capabilities of a client application and its proxyable target entry. When present in an entry, the Directory Server evaluates each operational attribute together to form a whitelist of potential users that can be proxied. If none of those attributes is present, then the user may potentially proxy as anyone.

The Directory Server supports a two-tier provision system that, when configured, can restrict specific users for proxied authorization. The first tier is a set of `ds-auth-may-proxy-as-*` operational attributes on the client entry that will bind to the server and carry out operations under the identity of another user. The second tier is a set of ds-auth-is-proxyable-* operational attributes on the user entry that defines whether access is allowed, prohibited, or required by means of proxied authorization. If allowed or required, the attributes define which client entries can proxy as the user.



```
uid=clientApp,dc=example,dc=com
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: uid=admin
```

```
uid=admin,dc=example,dc=com
ds-auth-is-proxyable: required
ds-auth-is-proxyable-by: uid=clientApp
```

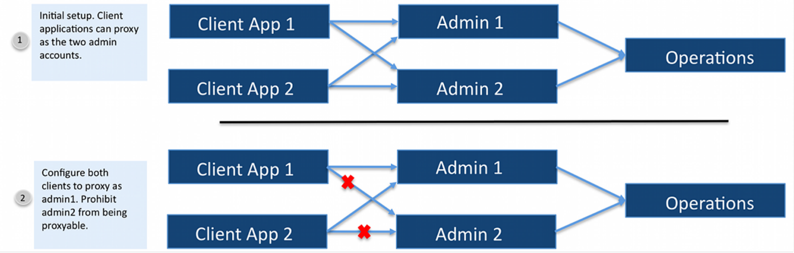
```
ldapsearch
-D uid=clientApp,dc=example,dc=com
-w password
-Y uid=admin,dc=example,dc=com
-b ou=People,dc=example,dc=com
“(objectclass=*)”
```

Figure 6: Proxying Operational Attributes

For example, if a client application, such as uid=clientApp, is requesting to search the ou=People,dc=example,dc=com branch as the user uid=admin, the command would look like this:

```
ldapsearch --bindDN uid=clientApp,dc=example,dc=com \
--bindPassword password \
--proxyAs uid=admin,dc=example,dc=com \
--baseDN ou=People,dc=example,dc=com \
"(object-class=*)
```

At bind, the Directory Server evaluates the list of users in the uid=clientApp entry based on the presence of any ds-auth-may-proxy-as-* attributes. In the figure below, the uid=clientApp entry has a ds-auth-may-proxy-as attribute with a value, uid=admin, which means that the client app user may proxy only as the uid=admin account. Next, the server confirms that uid=admin is in the list of proxyable users and then evaluates the ds-auth-is-proxyable-* attributes present in the uid=admin entry. These attributes determine the list of restricted users that either are allowed, prohibited, or required to proxy as the uid=admin entry. In this case, the uid=admin entry has the ds-auth-is-proxyable attribute with a value of "required", which indicates that the entry can only be accessed by means of proxied authorization. The uid=admin entry also has the ds-auth-is-proxyable-by attribute with a value of uid=clientApp, which indicates it can only be requested by the uid=clientApp entry. Once both sets of attributes have been confirmed, the uid=clientApp can bind to the server as the authenticated user. From this point, the Directory Server performs ACI evaluation on the branch to determine if the requested user has access rights to the branch. If the branch is accessible by the uid=clientApp entry, and then the search request is processed.


1 Initial setup. Client applications can proxy as the two admin accounts.

2 Configure both clients to proxy as admin1. Prohibit admin2 from being proxyable.

Figure 7: Proxying Operational Attributes Examples

About the ds-auth-may-proxy-as-* Operational Attributes

The Directory Server first evaluates the list of potential users that can be proxied for the authenticated user based on the presence of the ds-auth-may-* operational attributes in the entry. These operational attributes are multi-valued and are evaluated together if all are present in an entry:

- **ds-auth-may-proxy-as**. Specifies the user DNs that the associated user is allowed to proxy as. For instance, based on the previous example, you could specify in the uid=clientApp entry that it can proxy operations as uid=admin and uid=agent1.

  ```
  dn: uid=clientApp,ou=Applications,dc=example,dc=com
  ```

```
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: uid=admin,dc=example,dc=com
ds-auth-may-proxy-as: uid=agent1,ou=admins,dc=example,dc=com
```

- **ds-auth-may-proxy-as-group**. Specifies the group DNs and its group members that the associated user is allowed to proxy as. For instance, you could specify that the potential users that the `uid=clientApp` entry can proxy as are those members who are present in the group `cn=Agents,ou=Groups,dc=example,dc=com`. This attribute is multi-valued, so that more than one group can be specified. Nested static and dynamic groups are also supported.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as-group: cn=Agents,ou=Groups,dc=example,dc=com
```

- **ds-auth-may-proxy-as-url**. Specifies the DNs that are returned based on the criteria defined in an LDAP URL that the associated user is allowed to proxy as. For instance, the attribute specifies that the client can proxy as those entries that match the criteria in the LDAP URL. This attribute is multi-valued, so that more than one LDAP URL can be specified.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as-url: ldap:///ou=People,dc=example,dc=com??sub?
(l=austin)
```

About the ds-auth-is-proxyable-* Operational Attributes

After the Directory Server has evaluated the list of users that the authenticated user can proxy as, the server checks to see if the requested authorized user is in the list. If the requested authorized user is present in the list, then the server continues processing the proxable attributes in the entry. If the requested authorized user is not present in the list, the bind will fail.

The operational attributes on the proxying entry are as follows:

- **ds-auth-is-proxyable**. Specifies whether the entry is proxyable or not. Possible values are: "allowed" (operation may be proxied as this user), "prohibited" (operations may not be proxied as this user), "required" (indicates that the account will not be allowed to authenticate directly but may only be accessed by some form of proxied authorization).
- **ds-auth-is-proxyable-as**. Specifies any users allowed to use this entry as a target of proxied authorization.
- **ds-auth-is-proxyable-as-group**. Specifies any groups allowed to use this entry as a target of proxied authorization. Nested static and dynamic groups are also supported.
- **ds-auth-is-proxyable-as-url**. Specifies the LDAP URLs that are used to determine any users that are allowed to use this entry as a target of proxied authorization.

Restricting Proxied Authorization for Specific Users

To illustrate how the proxied authorization operational attributes work, it is best to set up a simple example where two LDAP clients, `uid=clientApp1` and `uid=clientApp2` can freely proxy two administrator accounts, `uid=admin1` and `uid=admin2`. We will add the `ds-auth-may-proxy-as-*` and the `ds-auth-is-proxyable-*` attributes to these entries to restrict how each account can use proxied authorization. For example, the two client applications will continue to proxy the `uid=admin1` account but the `uid=admin2` account will no longer be able to be used as a proxied entry.

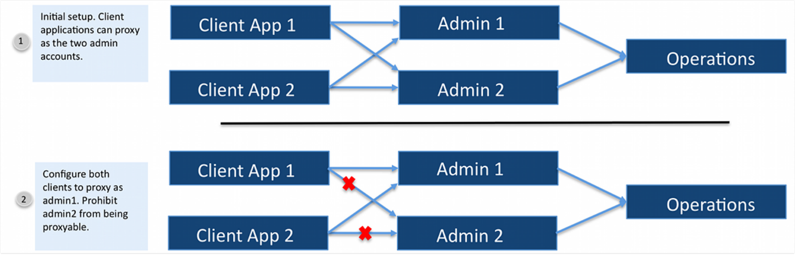

1
Initial setup. Client applications can proxy as the two admin accounts.
Client App 1
Client App 2
Admin 1
Admin 2
Operations
2
Configure both clients to proxy as admin1. Prohibit admin2 from being proxyable.
Client App 1
Client App 2
Admin 1
Admin 2
Operations

Figure 8: Proxy Users Example Scenario

To Restrict Proxied Authorization for Specific Users

1. For this example, set up two user entries, `uid=clientApp1` and `uid=clientApp2`, which will be proxying the `uid=admin1` and `uid=admin2` accounts to access the `ou=People,dc=example,dc=com` subtree. Both entries have the `proxied-auth` privilege assigned to it. Open a text editor and create an LDIF file. Add the file using the `ldapmodify` tool. Note that "..." indicates that other attributes present in the entry are not included in the example for readability purposes.

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth

dn: uid=clientApp2,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
```

2. Next, assign the ACI for each client application to the subtree, `ou=People,dc=example,dc=com`. Note that the ACIs should be on one line of text. The example displays the ACIs over multiple lines for readability.

```
dn: ou=People,dc=example,dc=com
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
       userdn="ldap:///uid=clientApp1,ou=Applications,dc=example,dc=com";)
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
       userdn="ldap:///uid=clientApp2,ou=Applications,dc=example,dc=com";)
```

3. Run a search for each entry. In this example, assume that there are two admin accounts: `admin1` and `admin2` that have full access rights to user attributes. You should be able to proxy as the `uid=admin1` and `uid=admin2` entries to access the subtree for both clients.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin1,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"

$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

4. Next, limit the proxied authorization capabilities for each client application. Update the `uid=clientApp1` entry to add the `ds-auth-may-proxy-as` attribute. In this example, the `ds-auth-may-proxy-as` attribute specifies that `uid=clientApp1` can proxy as the `uid=admin1` entry. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`. Note that `ds-auth-may-proxy-as` is multi-valued:

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
```

```
changetype: modify
add: ds-auth-may-proxy-as
ds-auth-may-proxy-as: uid=admin1,dc=example,dc=com
```

5. Repeat the previous step for the `uid=clientApp2` entry, except specify the `ds-auth-may-proxy-as-url`. The client entry may proxy as any DN that matches the LDAP URL.

   ```
   dn: uid=clientApp2,ou=Applications,dc=example,dc=com
   changetype: modify
   add: ds-auth-may-proxy-as-url
   ds-auth-may-proxy-as-url: ldap:///dc=example,dc=com??sub?(uid=admin*)
   ```

6. Next, we want to create a group of client applications that has `uid=clientApp1` and `uid=clientApp2` as its `uniquemembers` to illustrate the use of the `ds-auth-proxyable-by-group` attribute. In this example, set up a static group using the `groupOfUniqueNames` object class.

   ```
   dn: ou=Groups,dc=example,dc=com
   objectClass: top
   objectClass: organizationalunit
   ou: groups

   dn: cn=Client Applications,ou=Groups,dc=example,dc=com
   objectClass: top
   objectClass: groupOfUniqueNames
   cn: Client Applications
   ou: groups
   uniquemember: uid=clientApp1,ou=Applications,dc=example,dc=com
   uniquemember: uid=clientApp2,ou=Applications,dc=example,dc=com
   ```

7. Update the `uid=admin1` entry to provide the DN that it may be proxied as. Add the `ds-auth-is-proxyable` and the `ds-auth-is-proxyable-by` attributes. For instance, we make the `uid=admin1` a required proxyable entry, which means that it can only be accessed by some form of proxied authorization. Then, specify each DN that can proxy as `uid=admin1` using the `ds-auth-is-proxyable-by`. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`. Note that the example includes all three types of `ds-auth-is-proxable-by-*` attributes as an illustration, but, in an actual deployment, only one type of attribute is necessary if they all target the same entries.

   ```
   dn: uid=admin1,dc=example,dc=com
   changetype: modify
   add: ds-auth-is-proxyable
   ds-auth-is-proxyable: required
   -
   add: ds-auth-is-proxyable-by
   ds-auth-is-proxyable-by: ou=clientApp1,ou=Applications,dc=example,dc=com
   ds-auth-is-proxyable-by: ou=clientApp2,ou=Applications,dc=example,dc=com
   -
   add: ds-auth-is-proxyable-by-group
   ds-auth-is-proxyable-by-group: cn=Client
    Applications,ou=Groups,dc=example,dc=com
   -
   add: ds-auth-is-proxyable-by-url
   ds-auth-is-proxyable-by-url: ldap:///ou=Applications,dc=example,dc=com??sub?
   (uid=clientApp*)
   ```

8. Next, prohibit proxying for the `uid=admin2` entry by setting the `ds-auth-is-proxyable` to prohibited. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`.

   ```
   dn: uid=admin2,dc=example,dc=com
   changetype: modify
   add: ds-auth-is-proxyable
   ds-auth-is-proxyable: prohibited
   ```

9. Run a search using the proxied account. For example, run a search first with `uid=clientApp1` or `uid=clientApp2` that proxies as `uid=admin1` to return a successful operation. However, if you run a search

for uid=clientApp1 that proxies as uid=admin2, as seen below, you will see an "authorization denied" message due to uid=admin2 not matching the list of potential entries that can be proxied. The ds-auth-may-proxy-as-* attributes specify that the client can only proxy as uid=admin1:

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

```
One of the operational attributes (ds-auth-may-proxy-as,
ds-auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry
'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow
that user to be proxied as user 'uid=admin2,dc=example,dc=com'

Result Code:  123 (Authorization Denied)

Diagnostic Message:  One of the operational attributes (ds-auth-may-proxy-
as,
ds-auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry
'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow that
user to be proxied as user 'uid=admin2,dc=example,dc=com'
```

10. Run another search using uid=clientApp2, which attempts to proxy as uid=admin2. You will see an "authorization denied" message due to the presence of the ds-auth-is-proxyable:prohibited operational attribute, which states that uid=admin2 is not available for proxied authorization.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

```
The 'ds-auth-is-proxyable' operational attribute
in user entry 'uid=admin2,dc=example,dc=com' indicates that
user may not be accessed via proxied authorization

Result Code:  123 (Authorization Denied)

Diagnostic Message: The 'ds-auth-is-proxyable' operational
attribute in user entry 'uid=admin2,dc=example,dc=com' indicates
that user may not be accessed via proxied authorization
```

Working with Parameterized ACIs

The Directory Server supports the use of parameterized ACIs to control access to subtrees with homogenous administrative group or user patterns, which can be used in multi-tenant deployments. A single parameterized ACI can take the place of specifying identical ACIs on each tenant’s subtree. For example, the following parameterized ACI:

```
 (target="ldap:///o=($1),dc=example,dc=com")(version 3.0; acl \
"Subtree Admin Group members may search for and read entries in their
 subtree."; allow \
(search, read) groupdn="ldap:///cn=Subtree Admin
 Group,ou=groups,o=($1),dc=example,dc=com";)
```

Enables:

- Members of a group with DN "cn=Subtree Admin Group,ou=groups,o=Customers, dc=example,dc=com" to search for and read entries in the "o=Customers, dc=example,dc=com" subtree.
- Members of a group with DN "cn=Subtree Admin Group,ou=groups,o=Partners, dc=example,dc=com" to search for and read entries in the "o=Partners, dc=example,dc=com" subtree

The same access is granted for any substitution value for the ($1) parameter variable. If an operation tried to read the uid=user.1,o=acme,dc=example,dc=com entry, this ACI would be considered. This ACI would allow a read action, if the operation's user is a member of the cn=Subtree Admin Group,ou=groups,o=acme,dc=example,dc=com group.

Attribute values from the target DN can be replaced with different variables ($#) and then reference those variables in the group DN or user DN. The string representation of a parameter variable is constructed as follows:

- an open parenthesis
- a dollar sign
- a positive integer
- a closing parenthesis

In another example:

```
"population=($2),ou=Populations,environment=($1),ou=Environments,o=Acme"
```

The ($2) variable is the population ID in the DN of the target entry, and ($1) is the environment ID in the DN of the target entry. Those values from the target entry's DN are then substituted into the group DN or user DN value.

Parameter variables present in a parameterized ACI's target will be associated with the actual values from the resource DN. Each actual value will be substituted for its respective parameter variable in the ACI's target, and group bind rule DNs when performing access control on the resource entry. Parameter variables can be used in multiple RDNs in a parameterized target. A given RDN may have at most one parameter variable as its attribute value, and a given parameter variable may appear only once in the parameterized target.

The following values are examples of valid parameterized target DNs:

- ou=($1),dc=example,dc=com
- population=($2),ou=Populations,environment=($1),ou=Environments,o=Acme
- o=($1) (for a global ACI)

An ACI on an entry can only apply to that entry's subtree. If an ACI with a parameterized target is stored on an entry, that entry's DN must appear in a non-parameterized form as the rightmost RDNs of the parameterized target's DN. For example, if an ACI with a parameterized target were stored on the dc=example,dc=com entry, that parameterized target must end in dc=example,dc=com in a non-parameterized form. Global ACIs do not have this restriction. Each global ACI can have parameter variables in any or all of its parameterized target's RDNs. Additional restrictions for parameterized targets include:

- They may not be pattern ACIs. That is, they may not contain wildcards ('*').
- RDNs that are parameterized must be single-valued. For example, a given parameterized RDN may not consist of two or more type-value pairs joined by '+'.

Chapter

17

Managing the Schema

Topics:

This chapter presents a basic summary of the supported schema components on the PingDirectory Server and procedures to extend the schema with new element definitions. The chapter presents the following topics:

About the Schema

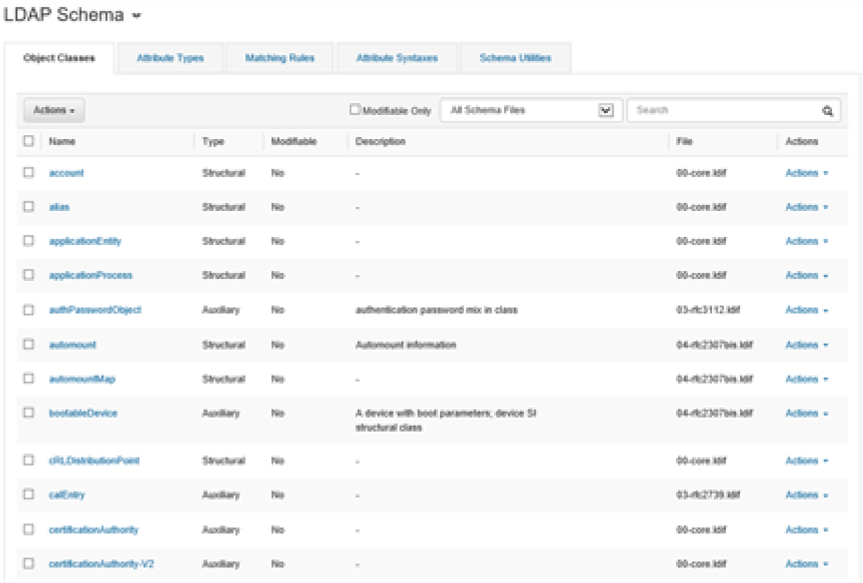
A schema is the set of directory server rules that define the structures, contents, and constraints of a Directory Information Tree (DIT). The schema guarantees that any new data entries or modifications meet and conform to these predetermined set of definitions. It also reduces redundant data definitions and provides a uniform method for clients or applications to access its Directory Server objects.

The PingDirectory Server ships with a default set of read-only schema files that define the core properties for the Directory Server. The Administrative Console provides a Schema Editor that administrators can use to view existing schema definitions and add new custom schema elements to their DIT. Any attempt to alter a schema element defined in a read-only file or add a new schema element to a read-only file will result in an "Unwilling to Perform" result.

About the Schema Editor

The Administrative Console Manangement Console provides a user-friendly graphical editor with tabs to manage any existing schema component related to the DIT: object classes, attributes, matching rules, attribute syntaxes, and schema utilities. The **Object Classes** and **Attribute Types** tabs enable viewing existing definitions as well as adding, modifying, or removing custom schema elements.

The **Matching Rules** and **Attribute Syntaxes** tabs are read-only and provide a comprehensive listing of all of the elements necessary to define new schema elements. The **Schema Utilities** tab provides a schema validator that allows you to load a schema file or perform a cut-and-paste operation on the schema definition to verify that it meets the proper schema and ASN.1 formatting rules. The **Utilities** tab also supports schema file imports by first checking for proper syntax compliance and generating any error message if the definitions do not meet specification.



□	Name	Type	Modifiable	Description	File	Actions
□	account	Structural	No	-	00-core.ldif	Actions ▾
□	alias	Structural	No	-	00-core.ldif	Actions ▾
□	applicationEntity	Structural	No	-	00-core.ldif	Actions ▾
□	applicationProcess	Structural	No	-	00-core.ldif	Actions ▾
□	authPasswordObject	Auxiliary	No	authentication password mix in class	03-rfc3112.ldif	Actions ▾
□	automount	Structural	No	Automount information	04-rfc2307bis.ldif	Actions ▾
□	automountMap	Structural	No	-	04-rfc2307bis.ldif	Actions ▾
□	bootableDevice	Auxiliary	No	A device with boot parameters; device SI structural class	04-rfc2307bis.ldif	Actions ▾
□	cRLDistributionPoint	Structural	No	-	00-core.ldif	Actions ▾
□	calEntry	Auxiliary	No	-	03-rfc2739.ldif	Actions ▾
□	certificationAuthority	Auxiliary	No	-	00-core.ldif	Actions ▾
□	certificationAuthority-V2	Auxiliary	No	-	00-core.ldif	Actions ▾

Figure 9: Example Schema Editor Screen

The Schema Editor provides two views for each definition: **Properties View** and **LDIF View**. The **Properties View** breaks down the schema definition by its properties and shows any inheritance relationships among the attributes. The **LDIF View** shows the equivalent schema definition in ASN.1 format, which includes the proper text spacing required for each schema element.

Default Directory Server Schema Files

The PingDirectory Server stores its schema as a set of LDIF files for a Directory Server instance in the `<server-root>/config/schema` directory. The Directory Server reads the schema files in alphanumeric order at startup, so that the `00-core.ldif` file is read first, then `01-pwpolicy.ldif`, and then the rest of the files. Custom schema files should be named so that they are loaded in last. For example, custom schema elements could be saved in a file labelled `99-user.ldif`, which loads after the default schema files are read at startup.

The Directory Server then uses the schema definitions to determine any violations that may occur during add, modify, or import requests. Clients applications check the schema (i.e., matching rule definitions) to determine the assertion value algorithm used in comparison or search operations.

The default set of schema files are present at installation and should not be modified. Modifying the default schema files could result in an inoperable server.

The schema files have the following descriptions:

Table 33: Default Schema Files

Schema Files	Description
00-core.ldif	Governs the Directory Server's core functions.
01-pwpolicy.ldif	Governs password policies.
02-config.ldif	Governs the Directory Server's configuration.
03-changelog.ldif	Governs the Directory Server's change log.
03-rfc2713.ldif	Governs Java objects.
03-rfc2714.ldif	Governs Common Object Request Broker Architecture (CORBA) object references.
03-rfc2739.ldif	Governs calendar attributes for vCard.
03-rfc2926.ldif	Governs Server Location Protocol (SLP) mappings to and from LDAP schemas.
03-rfc2985.ldif	Governs PKCS #9 public-key cryptography.
03-rfc3112.ldif	Governs LDAP authentication passwords.
03-rfc3712.ldif	Governs printer services.
03-uddiv3.ldif	Governs web services registries of SOA components.
04-rfc2307bis.ldif	Governs mapping entities from TCP/IP and UNIX into X.500 entries.

Extending the Directory Server Schema

The PingDirectory Server stores its schema as LDIF files in the `<server-root>/config/schema` directory. At startup, the Directory Server reads the schema files once in alphanumeric order starting with `00-core.ldif` and ending with any custom schema definition files, such as `99-user.ldif` if present.

You can extend the schema to include additional customizations necessary for your Directory Server data using one of the following methods:

- **Using the Schema Editor**. This method is the easiest and quickest way to set up a schema definition and have it validated for the correct ASN.1 formatting. The Editor lets you define your schema properties, load your custom file, or perform a cut-and-paste operation on a new schema element. If any errors exist in the file, the Schema Editor generates an error message if the schema definitions do not pass compliance.
- **Using a Custom Schema File**. You can create a custom schema file with your new definitions using a text editor, save it as `99-user.ldif`, and then import the file using the Schema Editor or the `ldapmodify` tool. You must name the custom LDIF file with a high two-digit number prefix, so that the Directory Server will read the file AFTER the core schema files are read at startup. For example, you can name the file, `99-myschema.ldif`, etc. See the next section, *General Tips on Extending the Schema* to see the requirements for naming each file.
- **Using the Command Line**. If you have a small number of additions, you can extend the schema over LDAP and from the command line using the `ldapmodify` tool. The Directory Server writes the new schema changes to

a file 99-user.ldif in the <server-root>/config/schema directory. However, this method can be cumbersome as schema definitions require strict adherence to text spacing and white space characters.

General Tips on Extending the Schema

You should consider the following points when extending the schema:

- Never modify the default schema files as doing so could damage the Directory Server's processing capabilities.
- Define all attributes first before they can be used in an object class. If you are using the Schema Editor to add new schema elements, then you can use the Quick Add Attributes option when defining new objectclasses.
- Define the parent object class first before creating object classes that inherit from the parent.
- The schema file naming syntax requires that custom schema files must begin with exactly two digits followed by a non-digit character, followed by a zero or more characters and ending with ".ldif". Note that the two digits do not need to be followed by a dash ("-"). Any files that do not meet this criteria will be ignored and either a NOTICE or SEVERE_WARNING message will be logged.

  Any file in the <server-root>/config/schema directory with a name that starts with "." or with a name that ends with a tilde ('~'), ".swp", or ".tmp" will generate a NOTICE message indicating that temporary files will be ignored. Any other file that does not meet the naming criteria will generate a SEVERE_WARNING message indicating that it will be ignored.
- Define custom attributes and object classes in one file. Typically, this file will be the 99-user.ldif. You can specify a different file name to which the Directory Server writes using the X-SCHEMA-FILE element and the file name in the definition. For example:

```
add: attributeTypes attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.1
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example'
  X-SCHEMA-FILE '99-custom.ldif' )
```

- Pay special attention to the white space characters in the schema definitions, where WSP means zero or more space characters, and SP means one or more space characters. The LDIF specification states that LDIF parsers should ignore exactly one space at the beginning of each continuation line, since continuation lines must begin with a space character. Thus, if you define a new schema definition with each keyword on a separate continuation line, you should add two spaces before an element keyword to be safe. For example, the following attribute definition has two spaces before the keywords: NAME, SUP, and X-ORIGIN.

```
attributeTypes: ( 2.5.4.32 NAME 'owner' SUP distinguishedName X-ORIGIN 'RFC
 4519' )
```

- In a replicated topology, any new schema additions will be replicated to other replication servers to their respective Schema backend. The additions will be written to the file specified by the X-SCHEMA-FILE extension or written to 99-user.ldif if no file is specified.

Managing Attribute Types

An attribute type determines the important properties related to an attribute, such as specifying the matching and syntax rules used in value comparisons. An attribute description consists of an attribute type and a set of zero or more options. Options are short, case-insensitive text strings that differentiate between attribute descriptions. For example, the LDAPv3 specification defines only one type of option, the tagging option, which can be used to tag language options, such as cn;lang-de;lang-sp or binary data, such as userCertificate;binary. You can also extend the schema by adding your own attribute definitions.

Attributes have the following properties:

- Attributes can be user attributes that hold information for client applications, or operational attributes that are used for administrative or server-related purposes. You can specify the purpose of the attribute by the `USAGE` element.
- Attributes are multi-valued by default. Multi-valued means that attributes can contain more than one value within an entry. Include the `SINGLE-VALUE` element if the attribute should contain at most one value within an entry.
- Attributes can inherit properties from a parent attribute as long as they both have the same USAGE, and the child attribute has the same `SYNTAX` or its `SYNTAX` allows values which are a subset of the values allowed by the `SYNTAX` of the parent attribute. For example, the surname (`sn`) attribute is a child of the name attribute.

Attribute Type Definitions

New attribute types do not require server code extensions if the provided matching rules and attribute syntaxes are used in the definitions. Administrators can create new attributes using the Schema Editor, which stores the definition in a file in the `<server-root>/config/schema` directory. See *Extending the Directory Server Schema* for more information.

The formal specification for attribute types is provided in RFC 4512, section 4.1.2 as follows:

```
AttributeTypeDescription = "(" wsp; Left parentheses followed by a white space
numericoid                         ; Required numeric object identifier
[ sp "NAME" sp qdescrs ]           ; Short name descriptor as alias for the OID
[ sp "DESC" sp qdstring ]          ; Optional descriptive string
[ sp "OBSOLETE" ]                  ; Determines if the element is active
[ sp "SUP" sp oid ]                ; Specifies the supertype
[ sp "EQUALITY" sp oid ]           ; Specifies the equality matching rule
[ sp "ORDERING" sp oid ]           ; Specifies ordering matching rule
[ sp "SUBSTR" sp oid ]             ; Specifies substrings matching rule
[ sp "SYNTAX" sp oidlen ]          ; Numeric attribute syntax with minimum
 upper bound
                                   ;  length expressed in {num}
[ sp "SINGLE-VALUE" ]              ; Specifies if the attribute is single
 valued in
                                   ;  the entry
[ sp "COLLECTIVE" ]                ; Specifies if it is a collective attribute
[ sp "NO-USER-MODIFICATION" ]      ; Not modifiable by external clients
[ sp "USAGE" sp usage ]            ; Application usage
extensions wsp ")"                 ; Extensions followed by a white space and
 ")"

usage = "userApplications" /       ; Stores user data
  "directoryOperation" /           ; Stores internal server data
  "distributedOperation" /         ; Stores operational data that must be
 synchronized
                                   ; across servers
  "dSAOperation"                   ; Stores operational data specific to a
 server and
                                   ; should not be synchronized across servers
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /              ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the definition
"X-APPROX" /              ; Specifies the approximate matching rule
"X-ALLOWED-VALUE" /       ; Explicitly specifies the set of allowed values
"X-VALUE-REGEX" /         ; Specifies the set of regular expressions to compare
 against
                          ;  attribute values to determine acceptance
"X-MIN-VALUE-LENGTH" /    ; Specifies the minimum character length for
 attribute values
"X-MAX-VALUE-LENGTH" /    ; Specifies the maximum character length for
 attribute values
```

```
"X-MIN-INT-VALUE" /       ; Specifies the minimum integer value for the
 attribute
"X-MAX-INT-VALUE" /       ; Specifies the maximum integer value for the
 attribute
"X-MIN-VALUE-COUNT" /     ; Specifies the minimum number of allowable values
 for the
                          ;  attribute
"X-MAX-VALUE-COUNT" /     ; Specifies the maximum number of allowable values
 for the
                          ;  attribute
"X-READ-ONLY"             ; True or False. Specifies if the file that contains
 the
                          ;  schema element is marked as read-only in the
 server
                          ;  configuration.
```

Basic Properties of Attributes

The Basic Properties section displays the standard elements in schema definition.

Table 34: Basic Properties of Attributes

Attributes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the attribute and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Syntax	Specifies the attribute syntax used. For example, the `userPassword` attribute uses the User Password Syntax whereas the `authPassword` attribute uses the Authentication Password Syntax.
Parent	Specifies the schema definition’s parent or supertype if any. The analogous LDIF equivalent is "SUP".
Multivalued	Specifies if the attribute can appear more than once in its containing object class.
Required By Class	Specifies any object classes that require the attribute.
Allowed By Class	Specifies any object classes that can optionally use the attribute.
Value Restrictions	Specifies any restriction on the value of the attribute.

The Extra Properties section provides additional auxiliary information associated with the attribute.

Table 35: Basic Properties of Attributes

Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element. For example, NAME (’sn’ ’surname’).
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Stored in File	Specifies the schema file that stores the definition in the `<server-root>/config/schema` folder.

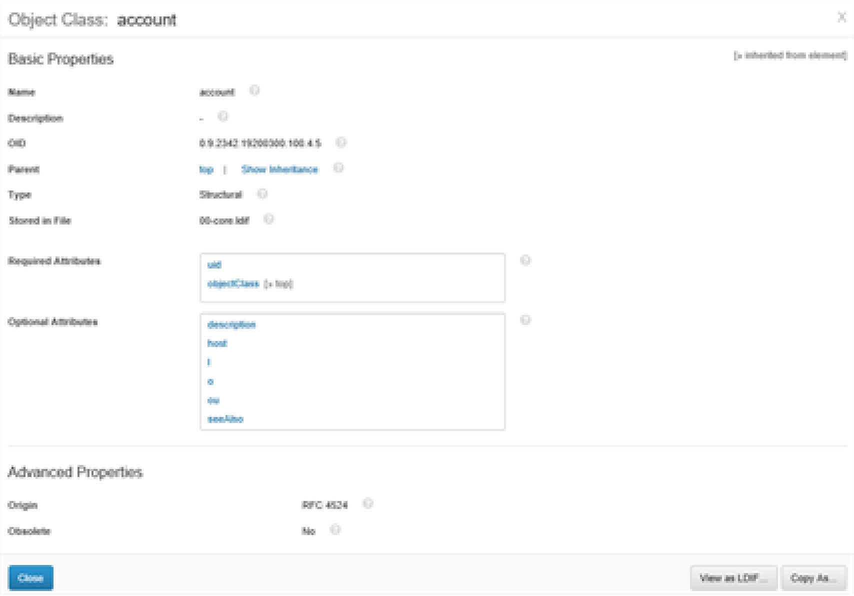
Attributes	Description
Usage	Specifies the intended use of the attribute. Choices are the following: userApplications directoryOperation distributedOperation dSAOperation
User-Modifiable	Specifies if the attribute can be modified by an authorized user.
Obsolete	Specifies if the schema definition is obsolete or not.
Matching Rules	Specifies the associated matching rules for the attribute.

Viewing Attributes

The Schema Editor displays all of the attribute types on your directory server instance. It shows the basic properties that are required elements plus the extra properties that are allowed within the attribute definition.

To View Attribute Types Using the Schema Editor

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is also running.
2. On the main menu, click **Schema**.
3. In the Administrative Console **Schema Editor**, click the **Attribute Types** tab.
4. Click a specific attribute to view its definition. In this example, click the `account` attribute. In the **Object Class** window, view the attribute properties.



Object Class: account

Basic Properties

[» inherited from element]

Name: account

Description: -

OID: 0.9.2342.19200300.100.4.5

Parent: top | Show Inheritance

Type: Structural

Stored in File: 00-core.ldif

Required Attributes: uid, objectClass [» top]

Optional Attributes: description, host, l, o, ou, seeAlso

Advanced Properties

Origin: RFC 4524

Obsolete: No

5. Click the **View as LDIF** button to see the equivalent attribute definition in ASN.1 format.

To View Attribute Types over LDAP

- Use `ldapsearch` to view a multi-valued operational attribute `attributeTypes`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" attributeTypes
```

To View a Specific Attribute Type over LDAP

- Use `ldapsearch` with the --dontWrap option and use the `grep` command to search for a specific attribute.

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base --dontWrap "(objectclass=*)" \
  attributeTypes | grep 'personalTitle'
```

Creating a New Attribute over LDAP

The following section shows how you can add the schema element from the previous section over LDAP. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting.

To Add an New Attribute to the Schema over LDAP

1. Create an LDIF file with the new attribute definition using a text editor. Save the file as `myschema.ldif`.

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ALLOWED-VALUES ’Y’ ’N’ ’y’ ’n’
  X-ORIGIN 'PingDirectory Server Example' )
```

2. Use `ldapmodify` to add the attribute.

```
$ bin/ldapmodify --filename myschema.ldif
```

3. Verify the addition by displaying the attribute using `ldapsearch`.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  --dontwrap "(objectclass=*)" attributeTypes | grep 'contractorStatus'
```

4. You can view the custom schema file at `<server-root>/config/schema/99-user.ldif`. You should see the following:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( contractorStatus-OID
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'PingDirectory Server Example' )
```

To Add Constraints to Attribute Types

- The Directory Server provides attribute type extensions that constrain the values for the associated attribute using the `DirectoryString` attribute syntax. The following schema definition includes two `attributeType` definitions for `myAttr1` and `myAttr2`. The first definition constrains the values for the attribute `myAttr1` to ’foo’, ’bar’, ’baz’. The second definition constrains the minimum allowable length for `myAttr2` to 1 and the maximum allowable length to 5.

```
attributeTypes: (1.2.3.4
  NAME ’myAttr1’
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ALLOWED-VALUES ( ’foo’ ’bar’ ’baz’ ))
attributeTypes: ( 1.2.3.5
  NAME ’myAttr2’
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-MIN-VALUE-LENGTH ’1’
  X-MAX-VALUE-LENGTH ’5’ )
```

Managing Object Classes

Object classes are sets of related information objects that form entries in a Directory Information Tree (DIT). The Directory Server uses the schema to define these entries, to specify the position of the entries in a DIT, and to control the operation of the server. You can also extend the schema by adding your own schema definitions.

Object classes have the following general properties:

- Object classes must have a globally unique name or identifier.
- Object classes specify the required and allowed attributes in an entry.
- Object classes can inherit the properties and the set of allowed attributes from its parent object classes, which may also be part of a hierarchical chain derived from the top abstract object class.
- Object classes that are defined in the PingDirectory Server can be searched using the `objectClasses` operational attribute. The Directory Server also has a special entry called the subschema subentry, which provides information about the available schema elements on the server.

Object Classes Types

Based on RFC 4512, object classes can be a combination of three different types:

- **Abstract object classes** are used as the base object class, from which structural or auxiliary classes inherit its properties. This inheritance is a one-way relationship as abstract object classes cannot be derived from structural or auxiliary classes. The most common abstract object class is `top`, which defines the highest level object class in a hierarchical chain of object classes.
- **Structural object classes** define the basic attributes in an entry and define where an entry can be placed in a DIT. All entries in a DIT belong to one structural object class. Structural object classes can inherit properties from other structural object classes and from abstract object classes to form a chain of inherited classes. For example, the `inetOrgPerson` structural object class inherits properties from the `organizationalPerson` structural class, which inherits from another object class, person.
- **Auxiliary object classes** are used together with structural object classes to define additional sets of attributes required in an entry. The auxiliary object class cannot form an entry alone but must be present with a structural object class. Auxiliary object classes cannot derive from structural object classes or vice-versa. They can inherit properties from other auxiliary classes and from abstract classes.

Object Class Definition

New object classes can be specified with existing schema components and do not require additional server code extensions for their implementation. Administrators can create new object classes using the Schema Editor, which manages schema in the `<server-root>/config/schema` directory. See *Extending the Directory Server Schema* for more information.

The object class definition is defined in RFC 4512, section 4.1.1, as follows::

```
ObjectClassDescription = "(" wsp; Left parenthesis followed by a white space
numericoid                    ; Required numeric object identifier
[ sp "NAME" sp qdescrs ]      ; Short name descriptor as alias for the OID
[ sp "DESC" sp qdstring ]     ; Optional descriptive string
[ sp "OBSOLETE" ]             ; Determines if the element is inactive
[ sp "SUP" sp oid ]                ; Specifies the direct superior object class
[ sp kind ]                        ; abstract, structural (default), auxiliary
[ sp "MUST" sp oids ]              ; Required attribute types
[ sp "MAY" sp oids ]               ; Allowed attribute type
extensions wsp ")"                 ; Extensions followed by a white space and
 ")"

usage = "userApplications" /       ; Stores user data
  "directoryOperation" /           ; Stores internal server data
  "distributedOperation" /         ; Stores operational data that must be
 synchronized
                                   ;   across servers
  "dSAOperation"                   ; Stores operational data specific to a
 server and
                                   ;   should not be synchronized across
 servers
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /               ; Specifies where the object class is defined
"X-SCHEMA-FILE" /          ; Specifies which schema file contains the definition
"X-READ-ONLY"              ; True or False. Specifies if the file that contains
                           ;   the schema element is marked as read-only
                           ;   in the server configuration.
```

Note: Although RFC 4512 allows multiple superior object classes, the PingDirectory Server allows at most one superior object class, which is defined by the SUP element in the definition.

Basic Object Class Properties

The Basic Properties section displays the standard elements in schema definition.

Table 36: Basic Properties of Attributes

Attributes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the object class and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Parent	Specifies the schema definition’s hierarchical parent or superior object class if any. An object class can have one parent. The analogous LDIF equivalent
Type	Specifies the type of schema definition: abstract, structural, or auxiliary. The analogous LDIF equivalent is "ABSTRACT", "STRUCTURAL", or "AUX
Required Attributes	Specifies any required attributes with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MUST".
Optional Attributes	Specifies any optional attributes that could be used with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MAY".

The Extra Properties section provides additional auxiliary information associated with the object class.

Table 37: Basic Properties of Attributes

Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element although most object classes do not have aliases.
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Obsolete	Specifies if the schema definition is obsolete or not.
Stored in File	Specifies the schema file that stores the definition in the `<server-root>/config/schema` folder.

Viewing Object Classes

You can view the object classes on your Directory Server by using the Administrative Console Schema Editor, over LDAP using the `ldapsearch` tool, or some third party tool. The Schema Editor displays all of the object classes on the directory server instance. It shows the basic properties that are required elements and the extra properties that are allowed within the object class.

To View Object Classes over LDAP

- Use `ldapsearch` tool to view a multi-valued operational attribute, `objectClasses`, which publishes the object class definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
--dontWrap "(objectclass=*)" objectClasses
```

```
dn: cn=schema
objectClasses: ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectClass X-ORIGIN 'RFC
 4512' )
objectClasses: ( 2.5.6.1 NAME 'alias' SUP top STRUCTURAL MUST
 aliasedObjectName
  X-ORIGIN 'RFC 4512' )
objectClasses: ( 2.5.6.2 NAME 'country' SUP top STRUCTURAL MUST c
  MAY ( searchGuide $ description ) X-ORIGIN 'RFC 4519' )
...(more output)...
```

Managing an Object Class over LDAP

The following section shows how you can manage an object class schema element over LDAP by adding a new attribute element to an existing object class. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting. The following example procedure adds an attribute, `contractorAddress`, to the custom schema file, then adds it to the `contractor` objectclass.

To Manage an Object Class over LDAP

1. Assume that you have defined the `contractorAddress` attribute, create an LDIF file called `contractorAddress-attr.ldif` with the following content:

```
dn: cn=schema
changetype: modify

add: attributeTypes attributeTypes: ( contractor-OID NAME
 'contractorAddress'
```

```
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'user defined'
  X-SCHEMA-FILE ’98-custom-schema.ldif’ )
  X-ORIGINS ’user defined’
  X-SCHEMA-FILE ’98-custom-schema.ldif’ )
```

2. Add the attribute using `ldapmodify`.

```
$ bin/ldapmodify --filename contractorAddress-attr.ldif
```

3. Next, create an LDIF file to modify the contractor objectclass to allow this attribute. When doing this, you are just re-adding the updated `objectClass` and the Directory Server will handle the proper replacement of the existing object class with the new one. Create a file called `contractor-oc.ldif`. Make sure that the lines are not wrapped, the `objectClasses` line should be one continuous line.

```
dn:cn=schema
changetype: modify
add: objectClasses
objectClasses: ( contractor-OID NAME ’contractor’
  DESC ’Contractor status information
  SUP top
  AUXILIARY MAY ( contractorStatus $ contractorAgency $ contractorAddress )
  X-ORIGIN ’Directory Server Example’
  X-SCHEMA-FILE ’98-custom-schema.ldif’ )
```

4. Update the `objectClass` using `ldapmodify` as follows:

```
$ bin/ldapmodify --filename contractor-oc.ldif
```

5. These schema changes will be replicated to all servers in the replication topology. Verify the change by looking at the `config/schema/98-custom-schema.ldif` file on the other servers in the replication topology to ensure that the changes are present.
6. If you need to add an index for this attribute, you can do so by using the `dsconfig` command-line utility. You will need to do this on each server in your topology unless you have server configuration groups set up. See *Configuring Server Groups* for more information.

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name contractorAddress --set index-type:equality
```

7. Rebuild the index online. This will not affect other indexes or entries since there is no currently existing data for this attribute on any entry.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index contractorAddress
```

Creating a New Object Class Using the Schema Editor

The procedures to create a new object class are similar to that of creating a new attribute. Make sure that any attributes that are part of the new object class are defined prior to defining the object class.

To Create a New Object Class Using the Schema Editor

1. Start the Administrative Console.
2. On the main menu, click **LDAP Schema**.
3. On the **Schema Editor**, click the **Object Classes** tab, and then click **New** located in the bottom left of the window.
4. Enter the properties for the new objectclass. In the **Attributes** box, filter the types of attributes required for the new object class. Click the right arrow to move it into the **Required** or the **Optional** box. All custom attributes appear at the bottom of the list in the **Attributes** box.

Extending the Schema Using a Custom Schema File

You can add new attributes and object classes to your Directory Server schema by creating a custom schema file. You can import the file using the Schema Editor, over LDAP using the `ldapmodify` tool, or from the command line. Make sure to define the attributes first, then define the object classes.

To Extend the Schema Using a Custom Schema File

1. Create an LDIF file with the new attribute extensions using a text editor.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example' )
attributeTypes: ( contractorAgency-OID NAME 'contractorAgency'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{256}
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'PingDirectory Server Example' )
```

2. In the same LDIF file, add a new object class definition after the attribute types. In this example, create an auxiliary object class, `contractor`, that alone cannot be used as an entry. The object class will be used to add supplemental information to the `inetOrgPerson` structural object class. The attributes are all optional for the new object class.

```
objectClasses: ( contractor-OID
  NAME 'contractor'
  DESC 'Contractor status information'
  SUP top
  AUXILIARY
  MAY ( contractorStatus $ contractorAgency )
  X-ORIGIN 'PingDirectory Server Example' )
```

3. Save the file as `99-auxobjclass.ldif`. and place it in the `<server-root>/config/schema` directory.
4. At this stage, the schema extensions are not loaded into the Directory Server yet. You have four options to load them:

   - Create a task that loads the new extensions into the schema. We create a task labelled with the ID "add-schema-99-auxobjclass" and add it using `ldapmodify`. The server does not need to be restarted using this method.

   ```
   dn: ds-task-id=add-schema-99-auxobjclass,cn=Scheduled Tasks,cn=tasks
   objectClass: top
   objectClass: ds-task
   objectClass: ds-task-add-schema-file
   ds-task-id: add-schema-99-auxobjclass
   ds-task-class-name:
    com.unboundid.directory.server.tasks.AddSchemaFileTask
   ds-task-schema-file-name: 99-auxobjclass.ldif
   ```

   - Import the schema file using the Administrative Console Schema Editor. You do not need to restart the server when using this method.
   - Place the `99-auxobjclass.ldif` file in the `<server-root>/config/schema` directory and restart the Directory Server. The schema file is read at startup.

- Add the schema file using `load-ldap-schema-file`. You do not need to restart the server when using this method.

  ```
  $ bin/load-ldap-schema-file --schemaFile config/schema 99-auxobjclass.ldif
  ```

5. Verify the addition by displaying the attribute using `ldapsearch`.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.9)" contractorStatus
```

```
dn: uid=user.9,ou=People,dc=example,dc=com
contractorStatus: TRUE
```

Managing Matching Rules

Matching rules determine how clients and servers compare attribute values during LDAP requests or operations. They are also used in evaluating search filter elements including distinguished names and attributes. Matching rules are defined for each attribute based on EQUALITY (e.g., two attributes are equal based on case, exact match, etc.), SUBSTR (e.g., assertion value is a substring of an attribute), and ORDERING (e.g., greater than or equal, less than or equal, etc.) properties.

Note: The PingDirectory Server supports an `APPROXIMATE` matching rule that compares similar attributes based on fuzzy logic. Thus, attributes that are similar or "sound-like" each other are matched. For example, "petersen" would match "peterson".

Matching Rule Definition

New matching rules require additional server code extensions to be implemented on the PingDirectory Server. If you need new matching rules, contact your authorized support provider for assistance.

The formal specification for attribute types is provided in RFC 4512, section 4.1.3 as follows:

```
MatchingRuleDescription = "(" wsp     ; Left parentheses followed by a white
 space
numericoid                            ; Required numeric object identifier
 identifying
                                      ;   this matching rule
[ sp "NAME" sp qdescrs                ; Short name descriptor
[ sp "DESC" sp qdstring               ; Description
[ sp "OBSOLETE" ]                     ; Specifies if the rule is inactive
sp "SYNTAX" sp numericoid             ; Assertion syntax
extensions wsp ")"                    ; Extensions followed by a white space
 and ")"
```

Default Matching Rules

The PingDirectory Server provides a large set of matching rules, which support a variety of applications. The default matching rules available for the Directory Server are listed in the table below for each matching rule type: Equality, Substring, Ordering, and Approximate matches.

Table 38: Default Matching Rules

Matching Rule/OID	Attribute Syntax/OID	Description
uuidMatch/ 1.3.6.1.1.16.2	UUID/ 1.3.6.1.1.16.1	Compares an asserted UUID with a stored UUID attribute value for equality. RFC 4530.
uuidOrderingMatch/ 1.3.6.1.1.16.3	UUID/ 1.3.6.1.1.16.1	Compares the collation order of an asserted UUID with a stored

Matching Rule/OID	Attribute Syntax/OID	Description
		UUID attribute value for ordering RFC 4530.
caseExactIA5Match/ 1.3.6.1.4.1.1466.109.114.1	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. RFC 4517.
caseIgnoreIA5Match/ 1.3.6.1.4.1.1466.109.114.2	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreIA5SubstringsMatch/ 1.3.6.1.4.1.1466.109.114.3	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
authPasswordExactMatch/ 1.3.6.1.4.1.4203.1.2.2	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password exact matching rule.
authPasswordMatch/ 1.3.6.1.4.1.4203.1.2.3	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password matching rule.
ds-mr-double-metaphone-approx/ 1.3.6.1.4.1.30221.1.4.1	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Syntax based on the phonetic Double Metaphone algorithm for approximate matching.
ds-mr-user-password-exact/ 1.3.6.1.4.1.30221.1.4.2	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password exact matching rule.
ds-mr-user-password-equality/ 1.3.6.1.4.1.30221.1.4.3	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password equality matching rule.
historicalCsnOrderingMatch/ 1.3.6.1.4.1.30221.1.4.4	1.3.6.1.4.1.30221.1.3.5	Compares the collation order of a historical change sequence number with a historical CSN attribute value.
caseExactIA5SubstringsMatch/ 1.3.6.1.4.1.30221.1.4.902	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. RFC 4517.
compactTimestampMatch/ 1.3.6.1.4.1.30221.2.4.1	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compact Timestamp matching rule.
compactTimestampOrderingMatch/ 1.3.6.1.4.1.30221.2.4.2	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compares the collation order of a compact timestamp number with an attribute value of Compact Timestamp syntax.
objectIdentifierMatch/ 2.5.13.0	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Compares an asserted value with an attribute value of OID syntax. RFC 4517.
distinguishedNameMatch/ 2.5.13.1	DN/ 1.3.6.1.4.1.1466.115.121.1.12	Compares an asserted value with an attribute value of DN syntax. Spaces around commas

Matching Rule/OID	Attribute Syntax/OID	Description
		and semicolons are ignored. Spaces around plus and equal signs around RDN components are ignored. RFC 4517.
caseIgnoreMatch/ 2.5.13.2	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreOrderingMatch/ 2.5.13.3	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
caseIgnoreSubstringsMatch/ 2.5.13.4	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring value with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
caseExactMatch/ 2.5.13.5	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value of Directory String syntax. RFC 4517.
caseExactOrderingMatch/ 2.5.13.6	Directory String 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. RFC 3698
caseExactSubstringsMatch/ 2.5.13.7	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Directory String syntax. RFC 3698
numericStringMatch/ 2.5.13.8	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares an asserted value with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringOrderingMatch/ 2.5.13.9	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares the collation order of the asserted string with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringSubstringsMatch/ 2.5.13.10	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.

Matching Rule/OID	Attribute Syntax/OID	Description
caseIgnoreListMatch/ 2.5.13.11	Postal Address/ 1.3.6.1.4.1.1466.115.121.1.41	Compares an asserted value with an attribute value which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreListSubstringsMatch/ 2.5.13.12	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares the asserted substring with an attribute value, which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 3698.
booleanMatch/ 2.5.13.13	Boolean/ 1.3.6.1.4.1.1466.115.121.1.7	Compares an asserted boolean value with an attribute value of BOOLEAN syntax. Returns true if the values are both TRUE or both FALSE. RFC 3698.
integerMatch/ 2.5.13.14	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares an asserted value with an attribute value of INTEGER syntax. RFC 4517.
integerOrderingMatch/ 2.5.13.15	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares the collation order of the asserted integer with an attribute value of Integer syntax. Returns true if the attribute value is less than the asserted value. RFC 3698.
bitStringMatch/ 2.5.13.16	Bit String/ 1.3.6.1.4.1.1466.115.121.1.6	Compares an asserted Bit String value with an attribute value of Bit String syntax. RFC 4517.
octetStringMatch/ 2.5.13.17	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares an asserted value with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
octetStringOrderingMatch/ 2.5.13.18	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares the collation order of the asserted octet string with an attribute value of Octet String syntax. Zero precedes a one bit. Shorter strings precede longer strings. RFC 3698.
octetStringSubstringsMatch/ 2.5.13.19	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
telephoneNumberMatch/ 2.5.13.20	Telephone Number/ 1.3.6.1.4.1.1466.115.121.1.50	Compares an asserted value with an attribute value of Telephone Number syntax. RFC 4517.

Matching Rule/OID	Attribute Syntax/OID	Description
telephoneNumberSubstringsMatch/ 2.5.13.21	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted value with the substrings of an attribute value of Telephone Number String syntax. RFC 4517.
presentationAddressMatch/ 2.5.13.22	Presentation Address/ 1.3.6.1.4.1.1466.115.121.1.43	Compares an asserted value with an attribute value of Presentation Address syntax. RFC 4517.
uniqueMemberMatch/ 2.5.13.23	Name and Optional UID/ 1.3.6.1.4.1.1466.115.121.1.34	Compares an asserted value with an attribute value of Unique Member syntax. RFC 4517.
protocolInformationMatch/ 2.5.13.24	Protocol Information/ 1.3.6.1.4.1.1466.115.121.1.42	Compares an asserted value with an attribute value of Protocol Information syntax. RFC 4517.
generalizedTimeMatch/ 2.5.13.27	Generalized Time/ 1.3.6.1.4.1.1466.115.121.1.24	Compares an asserted value with an attribute value of Generalized Time syntax. RFC 4517.
generalizedTimeOrderingMatch/ 2.5.13.28	Generalized Time 1.3.6.1.4.1.1466.115.121.1.24	Compares the collation order of the asserted string with an attribute value of Generalized Time String syntax and case is ignored. RFC 4517.
integerFirstComponentMatch/ 2.5.13.29	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Equality matching rules for subschema attributes between an Integer syntax and the value syntax. RFC 4517.
objectIdentifierFirstComponentMatch/ 2.5.13.30	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Equality matching rules for subschema attributes between an OID syntax and the value syntax. RFC 4517.
directoryStringFirstComponentMatch/ 2.5.13.31	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted Directory String value with an attribute value of type SEQUENCE whose first component is mandatory and of type Directory String. Returns true if the attribute value has a first component whose value matches the asserted Directory String using the rules of caseIgnoreMatch. RFC 3698.
wordMatch/ 2.5.13.32	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted word with any word in the attribute value for equality RFC 3698.
keywordMatch/ 2.5.13.33	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with any keyword in the attribute value for equality. RFC 3698.

Basic Matching Rule Properties

The Properties section displays the standard elements in a matching rule schema definition.

Table 39: Basic Properties of Matching Rules

Attributes	Description
Name	Specifies the descriptive and unique name of the element.
Description	Specifies an optional definition that describes the matching rule. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Type	Specifies the type of type of matching rule: Equality, Ordering, Substring, or Approximate.
Syntax	Specifies the matching rule syntax.
Used by Attributes	Specifies any attributes that use the corresponding matching rule.

Viewing Matching Rules

You can view the matching rules on your Directory Server by using the Schema Editor on the Administrative Console, over LDAP using the `ldapsearch` tool, or some third party tool.

The Schema Editor displays all of the matching rules on your Directory Server instance. It shows the basic properties that are allowed within the matching rule.

To View Matching Rules Over LDAP

- Use `ldapsearch` to view a multi-valued operational attribute, `matchingRules`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" matchingRules
```

Managing Attribute Syntaxes

The attribute type definition has a `SYNTAX` element, or attribute syntax, that specifies how the data values for the attribute are represented. The syntax can be used to define a large range of data types necessary for client applications. An attribute syntax uses the Abstract Syntax Notation One (ASN.1) format for its definitions.

Attribute Syntax Definition

New attribute syntaxes require additional code to be implemented on the PingDirectory Server. If you need new syntax definitions, contact your authorized support provider for assistance.

The formal specification for attribute types is provided in RFC 4512, section 4.1.5 as follows:

```
SyntaxDescription = "(" wsp
numericoid              ; Object identifier
[ sp "DESC" sp qdstring ] ; Description
extensions wsp ")"      ; Extensions followed by a white space and ")"
```

Default Attribute Syntaxes

The PingDirectory Server supports a large set of Attribute Syntax rules for applications. The default Attribute Syntax rules available for the directory server are listed in the table below.

Table 40: Default Attribute Syntaxes

LDAP Syntax	OID	Description
UUID	1.3.6.1.1.16.1	128-bit (16 octets) Universally Unique Identifier (UUID) used for Uniform Resource Names as defined in RFC 4122. For example, a4028c1a-f36e-11da-ba1a-04112154bd1e.
Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3	Syntax for the AttributeTypeDescription rule based on RFC 4517.
Binary	1.3.6.1.4.1.1466.115.121.1.5	Strings based on Basic Encoding Rules (BER) or Distinguished Encoding rules (DER). For example, an X.509 digital certificate or LDAP messages are BER encoded.
Bit String	1.3.6.1.4.1.1466.115.121.1.6	Sequence of binary digits based on RFC 4517. For example, '0010111'B.
Boolean	1.3.6.1.4.1.1466.115.121.1.7	TRUE or FALSE.
Certificate	1.3.6.1.4.1.1466.115.121.1.8	BER/DER-encoded octet strings based on an X.509 public key certificate as defined in RFC 4523.
Certificate List	1.3.6.1.4.1.1466.115.121.1.9	BER/DER-encoded octet string based on an X.509 certificate revocation list as defined in RFC 4523.
Certificate Pair	1.3.6.1.4.1.1466.115.121.1.10	BER/DER-encoded octet string based on an X.509 public key certificate pair as defined in RFC 4523.
Country String	1.3.6.1.4.1.1466.115.121.1.11	Two character country code specified in ISO 3166. For example, US, CA, etc.
DN	1.3.6.1.4.1.1466.115.121.1.12	Distinguished name of an entry as defined in RFC4514.
Delivery Method	1.3.6.1.4.1.1466.115.121.1.14	Sequence of services in preference order by which an entity receives messages as defined in RFC4517. For example, videotext $ telephone.
Directory String	1.3.6.1.4.1.1466.115.121.1.15	String of one or more characters from the Universal Character Set (UCS) using UCS Transformation Format 8 (UTF-8) encoding of the string.
DIT Content Rule Description	1.3.6.1.4.1.1466.115.121.1.16	DITContentRuleDescription as defined in RFC4517.
DIT Structure Rule Description	1.3.6.1.4.1.1466.115.121.1.17	DITStructureRuleDesciption as defined in RFC4517.
Enhanced Guide	1.3.6.1.4.1.1466.115.121.1.21	Combination of attribute types and filter operators to be used to construct search filters as defined in RFC4517. For example, person#(sn $EQ)#oneLevel.
Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22	Fax telephone number on the public switched telephone network as defined in RFC4517.

LDAP Syntax	OID	Description
Fax	1.3.6.1.4.1.1466.115.121.1.23	Image generated using Group 3 fax process as defined in RFC4517.
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	String representing data and time as defined in RFC4517. YYYYMMDDHHMMSS[.|,fraction][(+|-HHMM)|Z] For example, 201103061032, 201103061032-0500, or 201103061032Z ("Z" indicates Coordinated Universal Time).
Guide	1.3.6.1.4.1.1466.115.121.1.25	Attribute types and filter operators as defined in RFC4517.
IA5 String	1.3.6.1.4.1.1466.115.121.1.26	String of zero or more characters from the International Alphabet 5 (IA5) character set as defined in RFC4517.
Integer	1.3.6.1.4.1.1466.115.121.1.27	String representations of integer values. For example, the character string "1234" represents the number 1234 as defined in RFC4517.
JPEG	1.3.6.1.4.1.1466.115.121.1.28	Image in JPEG File Interchange Format (JFIF) as defined in RFC4517.
Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30	MatchingRuleDescription as defined in RFC4512.
Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31	Attribute types to which a matching rule is applied in an extensibleMatch search filter (RFC4511).
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	Distinguished name and an optional unique identifier that differentiates identical DNs as defined in RFC4517. For example, uid=jsmith,ou=People,dc=example,dc=com#'0111'B
Name Form Description	1.3.6.1.4.1.1466.115.121.1.35	NameFormDescription as defined in RFC4512.
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	Sequence of one or more numerals and spaces as defined in RFC4517. For example, 14 848 929 102.
Object Class Description	1.3.6.1.4.1.1466.115.121.1.37	ObjectClassDescription as defined in RFC 4512.
OID	1.3.6.1.4.1.1466.115.121.1.38	Object identifier as defined in RFC 4512.
Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39	Specifies an electronic mailbox as defined in RFC 4517. For example, otherMailbox = google $ user@gmail.com
Octet String	1.3.6.1.4.1.1466.115.121.1.40	Sequence of zero or more octets (8-bit bytes) as defined in RFC4517.
Postal Address	1.3.6.1.4.1.1466.115.121.1.41	Strings of characters that form a multi-line address in a physical mail system. Each component is separated by a "$". For example, 1234 Main St.$Austin, TX 78744$USA.
Protocol Information	1.3.6.1.4.1.1466.115.121.1.42	Undefined.

LDAP Syntax	OID	Description
Presentation Address	1.3.6.1.4.1.1466.115.121.1.43	String encoded OSI presentation address as defined in RFC 1278. For example, TELEX +00728722+RFC-1006+03+10.0.0.6
Printable String	1.3.6.1.4.1.1466.115.121.1.44	String of one or more printable ASCII alphabetic, numeric, and punctuation characters as defined in RFC 4517.
RFC3672 Subtree Specification	1.3.6.1.4.1.1466.115.121.1.45	Syntax based on subtree specification as defined as RFC 3672.
Supported Algorithm	1.3.6.1.4.1.1466.115.121.1.49	Octet string based on the LDAP-encoding for a supported algorithm value that results from the BER encoding of a SupportedAlgorithm ASN.1 value.
Telephone Number	1.3.6.1.4.1.1466.115.121.1.50	String of printable international telephone number representations in E.123 format as defined in RFC 4517. For example, +1 512 904 5525.
Teletex Terminal Identifier	1.3.6.1.4.1.1466.115.121.1.51	Identifier and telex terminal as defined in RFC 4517.
Telex Number	1.3.6.1.4.1.1466.115.121.1.52	String representing the telex number, country code, and answerback code as defined in RFC 4517. For example, 812374, ch, ehhg.
UTC Time	1.3.6.1.4.1.1466.115.121.1.53	Character string representing the data and time in UTC Time format as defined as RFC 4517: YYMMDDHHMM[SS][(+|-HHMM)|Z], where Z is the coordinated universal time. For example, 0903051035Z, 0903051035-0500.
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54	SyntaxDescription as defined in RFC4512.
Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58	Syntax for assertion values in an extensible match as defined in RFC 4517.
Authentication Password Syntax	1.3.6.1.4.1.4203.1.1.2	Encoded password storage syntax as defined in RFC 3112. For example, the syntax specifies the storage scheme in brackets as follows: <storage-scheme>$<auth component>$<auth value> For example: SSHA$xdEZWRqgyJk=$egDEFDXvdeeEnXUEIDPnd39dkpe=
User Password Syntax	1.3.6.1.4.1.30221.1.3.1	Encoded password storage syntax as defined in RFC 2307. For example, the syntax specifies the storage scheme in brackets as follows: {SSHA}XaljOF0ii3fOwCrU1klgBpWFayqSYs+5W1pMnw==
Relative Subtree Specification	1.3.6.1.4.1.30221.1.3.2	Similar to the RFC 3672 subtree specification except it uses an LDAP search filter as the specification filter.
Absolute Subtree Specification	1.3.6.1.4.1.30221.1.3.3	Syntax for a subset of entries in a subtree based on RFC 3672.
Sun-defined Access Control Information	1.3.6.1.4.1.30221.1.3.4	Syntax for access control instructions used in Sun Directory Servers.

LDAP Syntax	OID	Description
Compact Timestamp	1.3.6.1.4.1.30221.2.3.1	Syntax based on compact timestamp ISO 8601 format. For example, 20110306T102532.

Basic Attribute Syntax Properties

The Properties section displays the standard elements in an attribute syntax.

Table 41: Basic Properties of Attribute Syntaxes

Attributes	Description
Name	Specifies the descriptive and unique name of the element.
Description	Indicates an optional definition that describes the attribute syntax. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition.
Used by Attributes	Indicates any attributes that use the corresponding attribute syntax.

Viewing Attribute Syntaxes

You can view the attribute syntaxes on your Directory Server by using the Schema Editor on the Administrative Console, over LDAP using the `ldapsearch` tool, or some third party tool.

The Schema Editor displays all of the attribute syntaxes on your Directory Server instance. It shows the properties that are allowed within the attribute syntax.

To View Attribute Syntaxes Over LDAP

- Use `ldapsearch` to view the Directory Server's published list of attribute syntaxes using the multi-valued operational attribute, `ldapSyntaxes`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base "(objectclass=*)" ldapSyntaxes
```

Using the Schema Editor Utilities

The Schema Editor provides a **Schema Utilities** tab that enables importing new schema elements from a file and to checking schema compliance. If importing a schema file, the system automatically checks for compliance prior to the import. If the definition does not meet schema compliance, the system will display an error message. However, it is good practice to first check if your file is compliant with your schema prior to importing it.

To Check Schema Compliance Using the Schema Editor

1. Start the Administrative Console.
2. On the main menu, click **LDAP Schema**.
3. On the Schema Editor, click the **Schema Utilities** tab.
4. Click **Import Schema Elements** to read in an LDIF file, or copy-and-paste a new schema definition, and then click **Validate Entries**. If there is a problem, an error will be generated.

Modifying a Schema Definition

The Directory Server only allows schema definitions that are read-write to be edited. Schema elements indicated by the **Modifiable** column in the Schema Editor's tables can be modified.

To Modify a Schema Definition

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is running.
2. On the main menu, click **Schema**.
3. On the Schema Editor, click the **Object Classes** tab.
4. Select the object class that you want to modify, and then click **Actions -> Edit**. The Edit dialog box appears.
5. Make your changes, and then click **OK**.

Deleting a Schema Definition

The Directory Server only allows schema definitions that are read-write to be deleted. In general, those schema definitions in the **Custom** folder of the Schema Editor can be removed from the system. You should make sure that the schema element is not currently in use.

To Delete a Schema Definition

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is also running.
2. On the main menu, click **Schema**.
3. On the **Schema Editor**, click the **Object Classes** tab.
4. Select the object class that you want to remove, and then click **Actions -> Delete**.
5. On the **Confirmation** dialog, click **Yes** if you are sure that you want to delete the schema element.

Schema Checking

The PingDirectory Server provides full support for parsing all schema elements and provides access to all of its components. By default, the Directory Server enables schema checking for all operations, especially when importing data to the server or when modifying entries using the `ldapmodify` tool. Any schema violations will generate an error message to standard output.

To View the Schema Checking Properties

- Use `dsconfig` to view the schema checking property.

```
$ bin/dsconfig get-global-configuration-prop \
  --property check-schema
```

To Disable Schema Checking

Although not recommended, you can use the `dsconfig` tool to disable the schema checking. This feature only applies to public backends. Schema checking is enforced on private backends, such as changes to the Configuration, Schema, Task, and others. An admin action alert will be generated when attempting to disable schema checking using `dsconfig` interactive or non-interactive mode. The alert provides alternatives to disabling schema checking.

Run the `dsconfig` command and specify the `set-global-configuration-prop` subcommand to disable the `check-schema` property.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set check-schema:false
```

The system generates an admin action alert providing alternate options to disabling schema checking. Press **Enter** to continue the process or following one of the suggested tasks:

```
One or more configuration property changes require administrative action or
confirmation/notification.

Those properties include:

  *   check-schema: Schema checking should only be disabled as a last
 resort
      since disabling schema checking harms performance and can lead to
      unexpected behavior in the server as well as the applications that
      access it. There are less severe options for addressing schema issues:


  1.  Update the data to conform to the server schema.

  2.  Modify the server schema to conform to the data. Contact support
 before
      modifying the server's default schema.

  3.  Change the single-structural-objectclass-behavior property to allow
      entries to have no structural object class or multiple structural
 object
      classes.

  4.  Change the invalid-attribute-syntax-behavior property to allow
 attribute
      values to violate their attribute syntax.

  5.  Change the allow-zero-length-values property of the Directory String
      Attribute Syntax configuration to allow attributes with this syntax
 to
      have a zero length value.

Continue?  Choose 'no' to return to the previous step (yes / no) [yes]:
```

Managing Matching Rule Uses

Matching Rule Use definitions map certain attribute types with a matching rule definition for extensible match filters. Extensible match filters allows clients to search using DN components, for example, (`ou:dn:=engineering`) or using an OID number, for example, (`cn:1.2.3.4:=Sam Carter`). The matching rule use attribute publishes those attribute types and matching rule combinations, which can be used in extensible match assertions.

Typically, you define a matching rule use that is not normally specified in the attribute type definition. You can create new matching rule uses from the existing schema definitions by adding a custom schema file in the `<server-root>/config/schema` directory.

Matching Rule Use Definitions

Matching Rule Use can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.4 as follows:

```
MatchingRuleUseDescription = "(" wsp
numericoid                  ; Object identifier
[ sp "NAME" sp qdescrs ]    ; Short name descriptor
[ sp "DESC" sp qdstring ]   ; Description
[ sp "OBSOLETE" ]            ; Specifies if the rule use is inactive
sp "APPLIES" sp oid          ; Attribute types
extensions wsp ")"           ; Extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the definition
"X-READ-ONLY"             ; True or False. Specifies if the file that contains
                          ;   the schema element is marked as read-only in the
                          ;   server configuration.
```

To View Matching Rule Uses

A matching rule use lists the attribute types that are suitable for use with an `extensibleMatch` search filter.

- Use `ldapsearch` to view the Directory Server's published list of matching rule uses using the multi-valued operational attribute, `matchingRuleUse`, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

  ```
  $ bin/ldapsearch --baseDN cn=schema --searchScope base \
    "(objectclass=*)" matchingRuleUse
  ```

Managing DIT Content Rules

DIT Content Rules provide a way to precisely define what attributes may be present in an entry, based on its structural object class, without specifically creating a new object class definition. The DIT content rules can define the mandatory and optional attributes that entries contain, the set of auxiliary object classes that entries may be part of, and any optional attributes from the structural and auxiliary object classes that are prohibited from being present in the entries.

DIT Content Rule Definitions

DIT Content Rules can be specified with existing schema components and do not require additional code for its implementation. On the PingDirectory Server, only one DIT Content Rule may be defined for an entry in the structural object class.

The formal specification for attribute types is provided in RFC 4512, section 4.1.6 as follows:

```
DITContentRuleDescription = "(" wsp
numericoid                   ; Object identifier of the structural object class
 the rule applies to
[ sp "NAME" sp qdescrs ]     ; Short name descriptor
[ sp "DESC" sp qdstring ]    ; Description
[ sp "OBSOLETE" ]            ; Specifies if the rule is inactive
[ sp "AUX" sp oids ]         ; List of allowed auxiliary object classes
[ sp "MUST" sp oids ]        ; List of required attributes
[ sp "MAY" sp oids ]         ; List of allowed attributes in the entry
[ sp "NOT" sp oids ]         ; List of prohibited attributes in the entry
extensions wsp ")"           ; Extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /              ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the definition
"X-READ-ONLY"             ; True or False. Specifies if the file that contains
                          ;   the schema element is marked as read-only in
                          ;   the server configuration.
```

To View DIT Content Rules

- Use `ldapsearch` to view a multi-valued operational attribute `dITContentRules`, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" dITContentRules
```

Managing Name Forms

Name Forms define how entries can be named based on their structural object class. Specifically, name forms specify the structural object class to be named, as well as the mutually-exclusive set of required and allowed attributes to form the Relative Distinguished Names (RDNs) of the entries. Each structural object class may be associated with at most one name form definition.

Name Form Definitions

Name Forms can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.7.2 as follows:

```
NameFormDescription = "(" wsp
numericoid                    ; object identifier
[ sp "NAME" sp qdescrs ]      ; short name descriptor
[ sp "DESC" sp qdstring ]     ; description
[ sp "OBSOLETE" ]             ; not active
sp "OC" sp oid                ; structural object class
sp "MUST" SP oids             ; attribute types
[ sp "MAY" sp oids ]          ; attribute types
extensions wsp ")"            ; extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /              ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the definition
"X-READ-ONLY"             ; True or False. Specifies if the file that contains
                          ;   the schema element is marked as read-only in
                          ;   the server configuration.
```

To View Name Forms

- Use `ldapsearch` to view a multi-valued operational attribute `nameForms`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base "(objectclass=*)"
 nameForms
```

Managing DIT Structure Rules

DIT Structure Rules define which entries may be superior or subordinate to other entries in the DIT. Together with name forms, DIT Structure Rules determine how RDNs are added together to make up distinguished names (DNs). Because DITs do not have a global standard and are specific to a company's implementation, each DIT structure rule associates a name form with an object class and specifies each structure rule with an integer rule identifier, instead of an OID number. The identifier defines its relationship, either superior or subordinate, to another object class. If no superior rules are specified, then the DIT structure rule applies to the root of the subtree.

DIT Structure Rule Definition

DIT Structure Rules can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.7.1 as follows:

```
DITStructureRuleDescription = "(" wsp
ruleid                        ; object identifier
[ sp "NAME" sp qdescrs ]      ; short name descriptor
[ sp "DESC" sp qdstring ]     ; description
[ sp "OBSOLETE" ]             ; specifies if the rule is inactive
sp "FORM" sp oid              ; OID or name form with which the rule is
 associated
[ sp "SUP" ruleids ]          ; Superior rule IDs
extensions wsp ")"            ; extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /               ; Specifies where the rule is defined
"X-SCHEMA-FILE" /          ; Specifies which schema file contains the definition
"X-READ-ONLY"              ; True or False. Specifies if the file that contains
                           ;   the schema element is marked as read-only in
                           ;   the server configuration.
```

To View DIT Structure Rules

- Use `ldapsearch` to view a multi-valued operational attribute `dITStructureRules`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

  ```
  $ bin/ldapsearch --baseDN cn=schema --searchScope base \
    "(objectclass=*)" dITStructureRules
  ```

Managing JSON Attribute Values

The PingDirectory Server supports a JSON object attribute syntax, which can be used for attribute types whose values are JSON objects. The syntax requires that each value of this type is a valid JSON object. The following is an example schema definition:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( jsonAttr1-OID NAME 'jsonAttr1' DESC 'test json attribute
 support' EQUALITY jsonObjectExactMatch SYNTAX 1.3.6.1.4.1.30221.2.3.4 USAGE
 userApplications )
objectClasses: ( jsonObjectClass-OID NAME 'jsonObjectClass' AUXILIARY MAY
 jsonAttr1 )
```

Note: The EQUALITY matching rule should always be specified as `jsonObjectExactMatch` in the schema definition. Using the `jsonObjectFilterExtensibleMatch` is not valid in this case.

Two matching rules are provided to filter the JSON object syntax: `jsonObjectExactMatch` and `jsonObjectFilterExtensibleMatch`.

The `jsonObjectExactMatch` equality matching rule is used in evaluating equality filters in search operations, and for matching performed against JSON object attributes for add, compare, and modify operations. It determines whether two values are logically-equivalent JSON objects. The field names used in both objects must match exactly (although fields may appear in different orders). The values of each field must have the same data types. String values are case-insensitive. The order of elements in arrays is considered significant. Substring or approximate matching is not supported.

The `jsonObjectFilterExtensibleMatch` matching rule can perform more powerful matching against JSON objects. The assertion values for these extensible matching filters should be JSON objects that express the constraints for the matching. These JSON object filters are described in detail in the Javadoc documentation for the LDAP SDK for Java. Although the LDAP SDK can facilitate searches with this matching rule, these searches can be issued through any LDAP client API that supports extensible matching.

The following are example searches using the `jsonObjectFilterExtensibleMatch` rule with available filter types.

Equals field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "equals", "field" : ["stuff", "onetype", "name"], "value" : "John Doe" })'
```

Contains field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "containsField", "field" : "age", "expectedType" : "number" })'
```

Greater than field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "greaterThan", "field" : "age", "value" : 26, "allowEquals" : true})'
```

Configuring JSON Attribute Constraints

The PingDirectory Server can define a number of constraints for the fields included in JSON objects stored in values of a specified attribute type. Constraints that can be placed on a JSON field include:

- Requiring values of the field to have a specified data type.
- Indicating whether the field is required or optional.
- Indicating whether the field can have multiple values in an array. If a field is permitted to have array values, restrictions can also be placed on the number of elements that can be present in the array.
- Indicating whether the field can have a value that is the null primitive as an alternative to values of the indicated data type.
- Restricting values of string fields to a predefined set of values, that match a given regular expression, or a specified length.
- Restricting values of numeric fields with upper and lower bounds.

Any existing data that doesn't match newly-defined JSON constraints can still be decoded and managed by the server. Only new entries are subject to the new constraints. Attempts to alter existing entries with non-compliant JSON objects may require fixing those objects to make them conform to the new constraints.

The two global configuration properties that define schema constraints for JSON objects are `create-json-attribute-constraints` and `create-json-field-constraints` in `dsconfig`. In `dsconfig` interactive under advanced settings, the menu options are JSON Attribute Constraints and JSON Field Constraints. Configuration properties for each include:

- **attribute-type**. The name or object identifier of the attribute type with which the definition is associated. This attribute type must have the JSON object syntax. This property will be the naming attribute for the configuration entry.
- **allow-unnamed-fields**. A boolean value that indicates whether JSON objects, used as the values of attributes of the associated type, can include fields that are not referenced in the `attribute-value-constraints`

object. If this is `false`, JSON objects will only be permitted to have the defined fields. If this is `true` (which is the default behavior), JSON objects are permitted to have fields that are not referenced, and no constraints are imposed on those fields.

Unless a schema definition is configured with `allow-unnamed-fields` set to `false`, it is only necessary to include information about fields whose values should be indexed or tokenized. However, it may be desirable to define other fields that are expected in order to ensure that clients will not be permitted to store invalid values. See the *Working with JSON Indexes* for information about indexing JSON attributes.

As with standard LDAP schema, JSON schema constraints are enforced for any changes made after the constraints are defined. If there are already JSON values in the data before a JSON schema is defined for that attribute type (or before changes are made), values that already exist may violate those constraints. JSON schema constraints will also be enforced for data provided in an LDIF import, so that entries containing JSON objects that violate these constraints will be rejected.

Table 42: JSON Field Constraints

Property	Description
`field`	Specifies the path to the target field as a string, with periods to separate levels of hierarchy. If any field name in the hierarchy itself includes a period, that period should be escaped with a backslash.
`value-type`	Specifies the expected data type for the target field. Values can include: • `any`. The target field can have any value. • `boolean`. The target field must have a value of true or false. • `integer`. The target field must have a number that can be exactly represented as an integer. • `null`. The target field must have a value of null. • `number`. The target field must have a value that represents a valid JSON number • `object`. The target field must have a value that represents a valid JSON object. The `allowed-fields` array may contain additional elements that define constraints for the fields that may be present in the object. • `string`. The target field must have a value that represents a valid JSON string.
`is-required`	Specifies whether the target field is required to be present. If it is present, its value must be either `true` or `false`. If it is absent, a default of `false` is assumed.
`is-array`	Indicates whether the target field can be an array. If the value can be an array, all of the elements of the array must be of the type specified in the `value-type` field. It can be present with a value that is one of the single strings listed below. If it is absent, a value of `prohibited` is assumed. Values include: • `required`. The target field must be an array with zero or more values of the specified type, and must not be a single value of the specified type. • `optional`. The target field may be an array with zero or more values of the specified type, or it may be a single value of the specified type. • `prohibited`. The target field must not be an array and may only be a single value of the specified type.
`allow-null-value`	Specifies whether the target field can have a value of `null` as an alternative to the specified `value-type`. If present, its value must be either `true` or `false`. If it is absent, a default of `false` is assumed. This has no effect if the value-type is `null`.
`allow-empty-object`	If the value of the target field is a JSON object (or an array of JSON objects), specifies whether empty objects are permitted. If present, the value must be either `true` (the object may have zero or more fields) or `false` (the object may have one or more fields). If it is absent, a default of `false` is assumed.

Property	Description
index-values	Specifies whether values of the target field should be indexed in backends. If present, the value must be either true (the field should be indexed) or false (the field should not be indexed). If it is absent, a default of false is assumed. If true, the value-type must be boolean, integer, null, or string.
index-entry-limit	Specifies the maximum number of entries in which a particular value may appear before the entry ID list for that value is no longer maintained. If present, the value must be an integer greater than or equal to one. If it is absent, the server will use the default index entry limit for the associated backend. This is only applicable if index-values is true.
prime-index	Specifies whether backends should prime the contents of the JSON index database into memory when they are opened. This is ignored if the backend's prime-all-indexes property has a value of true.
cache-mode	Specifies the cache mode to use for the contents of the JSON index database. If the value is not specified, the backend's default cache mode will be used. If a cache mode of cache-keys-only is configured, priming will only load the internal nodes into memory for the index. If no-caching is configured, no priming is performed for the index.
tokenize-values	Specifies whether values of the target field should be tokenized in backends. If present, the value is either true (field values should be tokenized) or false. If it is absent, a value of false is assumed. If true, the value-type must be string.
allowed-values	Specifies an explicit set of values that the target field is permitted to have. If present, the value must be an array of strings. Any attempt to use a value not in this array for the associated field is rejected. This can only be used with a value-type of string. If not present, any value can be used as long as it satisfies all other defined constraints.
allowed-value-regular-expression	Specifies a regular expression that the target field will be required to match. If present, the value must be a single string or an array of strings representing valid regular expressions (which may require escaping to represent in JSON). Any attempt to use a value that does not match one of the provided regular expressions will be rejected. This can only be used with a value-type of string. If not present, values are not required to match any regular expression.
minimum-numeric-value	Specifies the lower bound for values of the target field. If present, the value must be a single number (it can be an integer). Any attempt to use a value that is less than this number is rejected. This can only be used with a value-type of integer or number. If not present, no minimum numeric value applies.
maximum-numeric-value	Specifies the upper bound for values of the target field. If present, the value must be a single number (it can be an integer). Any attempt to use a value that is more than this number is rejected. This can only be used with a value-type of integer or number. If not present, no maximum numeric value applies.
minimum-value-length	Specifies the minimum number of characters allowed for values of the target field. If present, the value must be an integer. Any attempt to use a value with fewer characters than this number is rejected. This can only be used with a value-type of string. If not present, no minimum length applies.
maximum-value-length	Specifies the maximum number of characters allowed for values of the target field. If present, the value must be an integer. Any attempt to use a value with more characters than this number is rejected. This can only be used with a value-type of string. If not present, no maximum length applies.

Property	Description
minimum-value-count	Specifies the minimum number of elements that must be present in an array. If present, the value must be an integer. Any attempt to use an array with fewer elements is rejected. This can only be used with an is-array value of required or optional. If not present, no minimum array value count applies.
maximum-value-count	Specifies the maximum number of elements that must be present in an array. If present, the value must be an integer. Any attempt to use an array with more elements is rejected. This can only be used with an is-array value of required or optional. If not present, no maximum array value count applies.

Note: When writing JSON objects in a local DB backend, field names and JSON primitive values of null, true, and false are always tokenized. Integers are either tokenized or compacted using their two's complement representation (other numbers are stored using string representations). Array and object sizes are compacted, and their contents are compacted based on their data types. String values are tokenized that match a recognizable format, including:

- Dates and times in common generalized time and ISO 8601 formats.
- UUIDs in which the alphabetic characters are either all uppercase or all lowercase.
- Strings of at least 12 bytes that are a valid base64 encoding.
- Strings of at least 6 bytes that are a valid hexadecimal encoding, in which the alphabetic characters are either all uppercase or all lowercase.

To Add Constraints to JSON Attributes

- Use the dsconfig tool (interactive or command line) to create and configure JSON attribute constraints. The following is a sample command:

```
$ bin/dsconfig create-json-attribute-constraints \
  --attribute-type appjson \
  --set enabled:true \
  --set allow-unnamed-fields:false
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.verified \
  --set value-type:boolean \
  --set is-required:true \
  --set index-values:true \
  --set tokenize-values:false \
  --set allow-null-value:true
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.type \
  --set value-type:string \
  --set is-required:true \
  --set index-values:true \
  --set tokenize-values:true \
  --set allowed-value:home \
  --set allowed-value:other \
  --set allowed-value:work \
  --set allow-null-value:false \
  --set minimum-value-length:1
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.value \
  --set value-type:string \
  --set is-required:true \
  --set index-values:true \
  --set tokenize-values:true \
  --set prime-index:true \
  --set allow-null-value:true \
  --set maximum-value-length:256 \
  --set minimum-value-length:1 \
  --set allowed-value-regular-expression:[-_\+\.\w\d]+@\w+\.\w{2,5}
```

Chapter

18

Managing Password Policies

Topics:

The PingDirectory Server provides a flexible password policy system to assign, manage, or remove password policies for root and non-root users. The password policy contains configurable properties for password expiration, failed login attempts, account lockout and other aspects of password and account maintenance on the Directory Server. The Directory Server also provides a global configuration option and a per-user privilege feature that disables parts of the password policy evaluation for production environments that do not require a password policy.

This chapter presents the following topics:

Viewing Password Policies

Password policies enforce a set of rules that ensure that access to data is not compromised through negligent password practices. The PingDirectory Server provides mechanisms to create and maintain password policies that determine whether passwords should expire, whether users are allowed to modify their own passwords, or whether too many failed authentication attempts should result in an account lockout. Many other options are available to fully configure a password policy for your PingData Platform system.

The Directory Server provides three out-of-the-box password policies that can be applied to your entries or as templates for configuring customized policies. The Default Password Policy is automatically applied to all users although it is possible to use an alternate password policy on a per-user basis. The Root Password Policy is enforced for the default root user, which uses a stronger password storage scheme (PBKDF2 rather than the salted 256-bit SHA-2 scheme) and also requires that a root user provide his or her current password to select a new password.

The Secure Password Policy provides a more secure option than the default policy that makes use of a number of features, including password expiration, account lockout, last login time and last login IP address tracking, password history, and a number of password validators.

Caution:

Using the Secure Password policy as-is may notably increase write load in the server by requiring updates to password policy state attributes in user entries and/or requiring users to change passwords more frequently. In environments in which write throughput is a concern (including environments spread across multiple data centers requiring replication over a WAN), it may be useful to consider whether the policy should be updated to reduce the number of entry updates that may be required.

To View Password Policies

- You can view the list of password policies configured on the Directory Server using the `dsconfig` tool, in either interactive or non-interactive mode, or the Administrative Console. The following example demonstrates the process for obtaining a list of defined password policies in non-interactive mode:

```
$ bin/dsconfig list-password-policies
```

```
Password Policy      : Type : password-attribute : default-password-storage-
scheme
------------------------:---------:--------------------:------------------------------
Default Password Policy : generic : userpassword    : Salted 256-bit SHA-2
Root Password Policy : generic : userpassword    : PBKDF2
Secure Password Policy  : generic : userpassword       : PBKDF2
```

To View a Specific Password Policy

- Use `dsconfig` or the Administrative Console to view a specific password policy. In this example, view the Default Password Policy that applies to all uses for which no specific policy has been configured.

```
$ bin/dsconfig get-password-policy-prop \
  --policy-name "Default Password Policy"
```

```
Property                                  : Value(s)
------------------------------------------:--------------------------
description                               : -
password-attribute                        : userpassword
default-password-storage-scheme           : Salted SHA-1
deprecated-password-storage-scheme        : -
password-validator                        : -
account-status-notification-handler       : -
allow-user-password-changes               : true
password-change-requires-current-password : false
```

```
force-change-on-add                       : false
force-change-on-reset                     : false
password-generator                        : Random Password Generator
require-secure-authentication             : false
require-secure-password-changes           : false
min-password-age                          : 0s
max-password-age                          : 0s
max-password-reset-age                    : 0s
password-expiration-warning-interval      : 5d
expire-passwords-without-warning          : false
allow-expired-password-changes            : false
grace-login-count                         : 0s
lockout-failure-count                     : 0s
lockout-duration                          : 0s
lockout-failure-expiration-interval       : 0s
require-change-by-time                    : -
last-login-time-attribute                 : ds-pwp-last-login-time
last-login-time-format                    : -
previous-last-login-time-format           : -
idle-lockout-interval                     : 0s
password-history-count                    : 0s
password-history-duration                 : 0s
```

About the Password Policy Properties

The Directory Server provides a number of configurable properties that can be used to control password policy behavior.

Note: To view a description of each of the password policy properties, see the *Ping Identity Directory Server Configuration Reference* that is bundled with the PingDirectory Server.

Some of the most notable properties include:

- **allow-user-password-changes**. Specifies whether users can change their own passwords. If a user attempts to change his/her own password, then the server will consult this property for the user's password policy, and will also ensure that the access control handler allows the user to modify the configured password attribute.
- **default-password-storage-scheme**. Specifies the names of the password storage schemes that are used to encode clear-text passwords for this password policy.
- **enable-debug**. When enabled, is used to debug password policy interaction. This property should be used in addition to the server's debug framework with a relevant debug target.
- **force-change-on-add**. Specifies whether users are required to change their passwords upon first authenticating to the Directory Server after their account has been created.
- **force-change-on-reset**. Specifies whether users are required to change their passwords after they have been reset by an administrator. An administrator is a user who has the `password-reset` privilege and the appropriate access control instruction to allow modification of other users' passwords.
- **idle-lockout-interval**. Specifies the maximum length of time that an account may remain idle (the associated user does not authenticate to the server) before that user is locked out. For accounts that do not have a last login time value, the password changed time or the account creation time will be used. If that information is not available, then the user will not be allowed to authenticate. It is strongly recommended that the server be allowed to run for a period of time with last login time tracking enabled (i.e., values for both `last-login-time-attribute` and `last-login-time-format` properties) to ensure that users have a last login time before enabling idle account lockout.
- **lockout-duration**. Specifies the length of time that an account is locked after too many authentication failures. The value of this attribute is an integer followed by a unit of seconds, minutes, hours, days, or weeks. A value of 0 seconds indicates that the account must remain locked until an administrator resets the password.
- **lockout-failure-count**. Specifies the maximum number of times that a user may be allowed to attempt to bind with the wrong password before that user's account becomes locked either temporarily (in which case the account will automatically be unlocked after a configurable length of time) or permanently (in which case an administrator

must reset the user's password before the account may be used again). For example, if the value is set to `3`, the user is locked out after three failed attempts, even if a fourth attempt is made with the correct password.

- **max-password-age**. Specifies the maximum length of time that a user can continue to use the same password before he or she is required to choose a new password. The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w). A minimum length of time can also be specified before the user is allowed to change the password.
- **password-change-requires-current-password**. Specifies whether users must include their current password when changing their password. This applies for both password changes made with the password modify extended operation as well as simple modify operations targeting the password attribute. In the latter case, if the current password is required then the password modification must remove the current value and add the desired new value (providing both the current and new passwords in the clear rather than using encoded representations).
- **password-expiration-warning-interval**. Specifies the length of time before a user's password expires that he or she receives notification about the upcoming expiration (either through the password policy or password expiring response controls). The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w).
- **password-retirement-behavior**. Specifies the behavior of a password that is allowed a retirement period before becoming invalid. This setting may be used by application service accounts that require a transition period while updating passwords. This is disabled by default.
- **password-validator**. Specifies the names of the password validators that are used with the associated password storage scheme. The password validators are invoked when a user attempts to provide a new password, to determine whether the new password is acceptable.
- **require-secure-authentication**. Indicates whether users with the associated password policy are required to authenticate in a secure manner. This might mean either using a secure communication channel between the client and the server, or using a SASL mechanism that does not expose the credentials.
- **require-secure-password-changes**. Indicates whether users with the associated password policy are required to change their password in a secure manner that does not expose the credentials.

Modifying an Existing Password Policy

You can modify an existing password policy to suit your company's requirements.

To Modify an Existing Password Policy

- Use `dsconfig` (in interactive or non-interactive mode) or the Administrative Console to modify the configuration for any defined password policy. The following example sets some of the properties presented in the previous section for the default password policy using `dsconfig` in non-interactive mode:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "max-password-age:90 days" \
  --set "password-expiration-warning-interval:14 days" \
  --set "lockout-failure-count:3" \
  --set "password-history-count:6"
```

Creating a New Password Policy

You can create any number of password policies in the Directory Server using either the `dsconfig` tool (in interactive or non-interactive mode) or the Administrative Console.

To Create a New Password Policy

- Use `dsconfig` (in interactive or non-interactive mode) or the Administrative Console to create a new password policy. The following example demonstrates the process for creating a new policy using `dsconfig` in non-interactive mode.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Demo Password Policy" \
  --set "password-attribute:userpassword" \
  --set "default-password-storage-scheme:Salted 256-bit SHA-2" \
  --set "force-change-on-add:true" \
  --set "force-change-on-reset:true" \
  --set "password-expiration-warning-interval:2 weeks" \
  --set "max-password-age:90 days" \
  --set "lockout-duration:24 hours" \
  --set "lockout-failure-count:3" \
  --set "password-change-requires-current-password:true"
```

To Assign a Password Policy to an Individual Account

To indicate that a user should be subject to a particular password policy (rather than automatically inheriting the default policy), include the `ds-pwp-password-policy-dn` operational attribute in that user's entry with a value equal to the DN of the desired password policy for that user. This attribute can be explicitly included in a user's entry, or it can be generated by a virtual attribute, which makes it easy to apply a custom password policy to a set of users based on a flexible set of criteria.

1. Create a file (`assign.ldif`) with the following contents:

```
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password
 Policies,cn=config
```

2. Use `ldapmodify` to apply the modification to the user's entry.

```
$ bin/ldapmodify --filename assign.ldif
```

To Assign a Password Policy Using a Virtual Attribute

It is possible to automatically assign a custom password policy for a set of users using a virtual attribute. The virtual attribute can be configured so that it can use a range of criteria for selecting the entries for which the virtual attribute should appear.

1. Create an LDIF file, which may be used to add a group to the server.

```
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: Groups

dn: cn=Engineering Managers,ou=groups,dc=example,dc=com
objectClass: groupOfUniqueNames
objectClass: top
cn: Engineering Managers
uniqueMember: uid=user.0,ou=People,dc=example,dc=com ou: groups
```

2. Use `ldapmodify` to add the entries to the server.

```
$ bin/ldapmodify --defaultAdd --filename groups.ldif
```

3. Use `dsconfig` to create a virtual attribute that will add the `ds-pwp-pasword-policy-dn` attribute with a value of `cn=Demo Password Policy,cn=Password Policies,cn=config` to the entries for all users that are members of the `cn=Engineering Managers,ou=Groups,dc=example,dc=com` group.

```
$ bin/dsconfig create-virtual-attribute \
  --name "Eng Mgrs Password Policy" \
  --type user-defined \
  --set "description:Eng Mgrs Grp PWPolicy" \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "value:cn=Demo Password Policy,cn=Password Policies,cn=config" \
  --set "group-dn:cn=Engineering Managers,ou=Groups,dc=example,dc=com
```

4. Use `ldapsearch` to verify that a user in the group contains the assigned password policy DN.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
ds-pwp-password-policy-dn
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password
 Policies,cn=config
```

Deleting a Password Policy

You can delete a password policy using either the `dsconfig` tool (in interactive or non-interactive mode) or the Administrative Console.

To Delete a Password Policy

Custom password policies may be removed using either the `dsconfig` tool (in interactive or non-interactive mode) or the Administrative Console).

- Run the `dsconfig` command with the delete-password-policy subcommand to remove a password policy.

```
$ bin/dsconfig delete-password-policy \
  --policy-name "Demo Password Policy"
```

Modifying a User's Password

There are two primary ways to change user passwords in the Directory Server:

- Perform a modify operation which replaces the value of the password attribute (often `userPassword`). Note that in some configurations, when a user attempts to change his or her own password it may be necessary to perform the modification by removing the password value and adding the desired new value to demonstrate that the user knows the current password value.
- Use the password modify extended operation to change the password. Note that if a user is changing his or her own password, it may be necessary to provide the current password value. The server will allow a new password to be provided (assuming that the new password is acceptable to all configured password validators), or it may automatically generate a new password for the user.

Note that regardless of the mechanism used to change the password, all password values should be provided in clear text rather than pre-encoded, and the user will be required to have sufficient access control rights to update the password attribute in the target user's entry. Further, when one user attempts to change the password for another user, then the requester will be required to have the `password-reset` privilege.

Validating a Password

The requirements that the server will impose for a password change can be displayed to users. The get password quality requirements extended operation can be used to retrieve information about the requirements, which can then be sent to an end user before an attempted password change. These requirements can also be used to enable client-side validation, so that any password problems can be identified before it is sent to the server. The password validation details request control can be included in an add or modify request, or a password modify extended request, to identify which validation requirements were not met by the password provided in the request.

Password validators can be configured with user-friendly messages that describe the password requirements, and the messages that should be returned if a proposed password does not satisfy those requirements. The server will generate these messages if they are not provided in the configuration.

Password validator properties include `validator-requirement-description` and `validator-failure` message. The following is a simple password validator configuration that requires passwords to contain a minimum of five characters, and lists custom validator messages:

```
$ dsconfig create-password-validator \
  --validator-name "Minimum 5 Characters Password Validator" \
  --type length-based --set enabled:true \
  --set "validator-requirement-description:The password must contain
         at least 5 characters." \
  --set "validator-failure-message:The password did not contain
         at least 5 characters." \
  --set min-password-length:5
```

After the password validator is created, it should be assigned to a Password Policy to take affect:

```
$ dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Minimum 5 Characters Password Validator"
```

Retiring a Password

An account password can be retired and rotated out of service, instead of immediately invalidated. This enables a new password to be assigned to an account while keeping the original password valid for a period of time to enable a transition. This is useful for application service accounts that require uninterrupted authentication with the server.

This behavior is disabled by default, but can be enabled in the password policy configuration by setting the `password-retirement-behavior` and `maximum-retired-password-age` properties.

To manually retire an account password or purge a password that has been retired, use the `ldapmodify` and `ldappasswordmodify` commands with options `-- retireCurrentPassword` and `--purgeCurrentPassword`. To use these commands on an account, the password policy that governs the account must have the `password-retirement-behavior` enabled.

To Change a User’s Password using the Modify Operation

- Use `ldapmodify` to change a user’s password by replacing the `userPassword` attribute.

```
$ bin/ldapmodify
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: newpw
```

To Change a User’s Password using the Password Modify Extended Operation

- Use `ldappasswordmodify` to request that the Password Modify extended operation be used to modify a user’s password.

```
$ bin/ldappasswordmodify --authzID dn:uid=jdoe,ou=People,dc=example,dc=com \
  --newPassword newpw
```

To Use an Automatically-Generated Password

- Use `ldappasswordmodify` to automatically generate a new password for a user.

```
$ bin/ldappasswordmodify --authzID "u:user.1"
```

```
The LDAP password modify operation was successful
Generated Password: fbi27oqy
```

Enabling YubiKey Authentication

Users can be enabled to authenticate with YubiKey devices (available from Yubico), which generate secure one-time passwords, with the UNBOUNDID-YUBIKEY-OTP SASL mechanism. A YubiKey device generates a different password for every authentication attempt, and that one-time password is sent to a validation service to ensure that it is genuine and has not been used in an earlier authentication attempt. Although it is possible to use this one-time password as the only proof of identity, it is typically combined with a static password as a form of two-factor authentication.

YubiKey authentication requires server configuration and the addition of this capability to a user entry. Configuration of a client ID and API key to use when communicating with the validation service is also required. The API key is a shared secret between the YubiKey validation service and the client that is interacting with it, and is used when generating digital signatures so that both the server and the YubiKey validation service can ensure that the peer server is genuine.

All server and user entry configuration details are available in the *Security Guide*.

Enabling Social Login

Authentication involving credentials that do not reside in, or cannot be forwarded to or validated by the Directory Server (such as social login through Facebook, Google, or Twitter) can be enabled with the the UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION SASL mechanism. The bind request will not include any credentials, and authentication with this mechanism will not actually change the state of the underlying client connection. The server will behave as if the bind request included the retain identity request control, regardless of whether or not that control was included.

Bind requests using this mechanism can include any request controls that are permitted with other bind requests. If the externally-processed authentication is successful, the client can include the get password policy state issues request control in the bind request to obtain information about any password policy state issues that can cause the Directory Server authentication attempt to fail. The password policy request control can also be included to obtain certain password policy state warnings and errors, or to look for the password expired/password expiring controls in the bind response.

All server and user entry configuration details are available in the *PingDirectory Server Security Guide*.

Managing User Accounts

The Directory Server provides a user management utility, the `manage-account` tool, that provides a means to quickly view and manipulate a number of password and account policy properties for a user or group of users.

Note: A disabled account status (for example, `account-is-disabled: true`) is different from an *account lockout* due to password policy. Unlocking a user account can be done with the `manage-account` tool. A disabled account requires the administrator to enable the account; password resets are not involved.

PingDirectory Server also hosts the Self Service Account Manager project at https://github.com/pingidentity/ssam, which is a customizable web application allowing users to perform their own account registration, profile updates, and password changes. The project is for testing and development purposes, and is not a supported PingDirectory Server application.

To Return the Password Policy State Information

- Use `manage-account` to get information about the account's password policy.

```
$ bin/manage-account get-all \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Password Policy DN: cn=Demo Password Policy,cn=Password Policies,cn=config
```

```
Account Is Disabled: false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time: 19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration: 1209600
Seconds Until Password Expiration Warning: 0
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count: 3
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset: false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count: 0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

To Determine Whether an Account is Disabled

- Use `manage-account` to determine whether a user's account has been disabled.

```
$ bin/manage-account get-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: true
```

To Disable an Account

- Use `manage-account` to disable a user's account.

```
$ bin/manage-account set-account-is-disabled \
  --operationValue true --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: true
```

To Enable a Disabled Account

- Use `manage-account` to enable a user's account.

```
$ bin/manage-account clear-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: false
```

To Assign the Manage-Account Access Privileges to Non-Root Users

Non-root users (e.g., `uid=admin`) with admin right privileges require access control permission to interact with certain password policy operational attributes when using the `manage-account` tool.

For example, the presence of the `ds-pwp-account-disabled` operational attribute in an entry determines that the entry is disabled. If the non-root admin user does not have the access privilege to read or interact with the `ds-pwp-account-disabled` operational attribute, the `manage-account` tool may report that the account is active. An account is considered active if the `ds-pwp-account-disabled` operational attribute does not exist in the entry or if the admin user does not have permission to see it.

Use the following procedure to give access rights to the non-root admin user.

1. Create a non-root user admin account, such as `uid=admin,dc=example,dc=com`. Grant the `password-reset` privilege to the account. See step 1 and 6 in the Configuring Administrators section for more information.

2. Run the `manage-account` tool to view the account status for an account.

```
$ bin/manage-account get-all \
  --targetDN uid=user.0,ou=People,dc=example,dc=com
```

```
Password Policy DN:  cn=Default Password Policy,cn=Password
 Policies,cn=config
Account Is Disabled:  false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time:  19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:
Seconds Until Password Expiration Warning:
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset:  false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count:  0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

3. Grant access control privileges to an account. The following allows access to manage accounts to a helpdesk user. Depending on the configuration requirements, this user may also need the `permit-get-password-policy-state-issues` and `password-reset` privileges.

```
dn: dc=example,dc=com
 changetype: modify
 add: aci
 aci: (targetattr="userPassword||ds-pwp-last-login-time||ds-pwp-password-
changed-by-required-time||ds-pwp-reset-time||ds-pwp-warned-time||
 ds-pwp-account-disabled||ds-pwp-account-expiration-time||ds-pwp-password-
policy-dn||ds-pwp-auth-failure||ds-pwp-last-login-ip-address||
 ds-pwp-retired-password||ds-pwp-account-activation-time||pwdReset||
pwdChangedTime||pwdAccountLockedTime")
 (version 3.0; acl "Grant full access to PWP related attributes to
 helpdesk"; allow (all) userdn="ldap:///uid=helpdesk,dc=example,dc=com";)
```

4. Run the `manage-account` tool to disable an account. The following command sets the `account-is-disabled` property to true for the `uid=user.0,dc=example,dc=com`.

```
$ bin/manage-account set-account-is-disabled \
  --targetDN uid=user.0,ou=People,dc=example,dc=com \
  --operationValue true
```

```
Account Is Disabled:  true
```

5. Run the `ldapsearch` tool to view the presence of the `ds-pwp-account-disabled` operational attribute in the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" "+"
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-account-disabled: true
```

Disabling Password Policy Evaluation

The Directory Server provides a global configuration property (`disable-password-policy-evaluation`) that can be used to disable most password policy evaluation processing. This provides a convenience for those production environments that do not require password policy support. If the `disable-password-policy` property is set to true, passwords will still be encoded and evaluated, but only account expiration and account disabling will be in effect. All other password policy properties, such as password expiration, lockout, and force change on add or reset, are ignored.

The server also supports the use of a `bypass-pw-policy` privilege, which can be used to skip password policy evaluation for operations on a per-user basis. If a user has this privilege, then they will be allowed to perform operations on user entries that would normally be rejected by the password policy associated with the target entry. Note that this privilege will not have any effect for bind operations.

To Globally Disable Password Policy Evaluation

- Use `dsconfig` to set the `disable-password-policy-evaluation` property globally for the Directory Server.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set "disable-password-policy-evaluation:false"
```

To Exempt a User from Password Policy Evaluation

- Use `ldapmodify` to add the `bypass-pw-policy` privilege to a user entry.

```
$ bin/ldapmodify
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: bypass-pw-policy
```

Managing Password Validators

A password validator is a password policy component that is used to determine if a new password is acceptable. A password policy can be configured with any number of password validators. If a password policy is configured with multiple password validators, then all of them must consider a proposed new password acceptable before it will be allowed.

Password Validators

The PingDirectory Server offers a number of types of password validators, including those listed in the following table. It is also possible to use the Server SDK to create custom password validators with whatever constraints are necessary for your environment.

Table 43: Password Validators

Password Validators	Description
Attribute Value	Ensures that the proposed password does not match the value of another attribute in the user’s entry. The validator can be configured to look in all attributes or in a subset of attributes. It can perform forward and reverse mapping, and it can also reject values which are substrings of another attribute.
Character Set	Ensures that the proposed password contains a sufficient number of characters from one or more user-defined character sets. For example, the validator can

Password Validators	Description
	ensure that passwords must have at least one lowercase letter, one uppercase letter, one digit, and one symbol.
Commonly-Used Passwords Dictionary	Ensures that the proposed password is not one of 10,000 commonly used passwords. These are words that are common for attackers to use when trying to access user accounts. The Commonly-Used Passwords validator is invoked by the Secure Password Policy by default. The word list is located in `<server-root>/config/commonly-used-passwords.txt`, and can be used to create a custom validator, but should not be modified.
Dictionary	Ensures that the proposed password is not present in a specified dictionary file, optionally also testing the password with all characters in reverse order. A large dictionary file is provided with the server, but the administrator can supply an alternate dictionary. In this case, then the dictionary must be a plain-text file with one word per line.
Haystack Password Validator	Ensures that the proposed password is secure based on a combination of its length and the types of characters that it contains. For example, a longer password containing only lowercase letters may be stronger than a shorter password containing a mix of uppercase and lowercase letters, numbers, and symbols. This is based on the Gibson Research Corporation Password Haystacks concept.
Length-Based Password Validator	Ensures that the number of characters in the proposed new password is within an acceptable range. Both a maximum and minimum number of characters may be specified.
Regular Expression Validator	Ensures that a proposed password either matches or does not match a given regular expression.
Repeated Characters	Ensures that a proposed password does not contain a substring in which the same character is repeated more than a specified number of times (for example, "aaaaa" or "aaabbb"). The validator can be configured to operated in a case-sensitive or case-insensitive manner, and you can also define custom sets of equivalent characters (for example, you could define all digits as equivalent, so the proposed password could not contain more than a specified number of consecutive digits.
Similarity-Based Password Validator	Ensures that the proposed new password is not too similar to the current password, using the Levenstein Distance algorithm, which calculates the number of characters that need to be inserted, removed, or replaced to transform one string into another. Note that for this password validator to be effective, it is necessary to have access to the user's current password. Therefore, if this password validator is to be enabled, the `password-change-requires-current-password` attribute in the password policy configuration must also be set to true.
Unique Characters	Ensures that the proposed password contains at least a specified minimum number of unique characters, optionally using case-insensitive validation.

Configuring Password Validators

You can use the `dsconfig` configuration tool or the Administrative Console to configure or modify any password validators. Once you have defined your password validators, you can add them to an existing password policy. The following example procedures show the dsconfig non-interactive commands necessary to carry out such tasks. If you use `dsconfig` in interactive command-line mode, you can access the Password Validator menu in the Basic Objects menu. For more details on the password validator properties, see the *PingDirectory Server Configuration Reference*.

To View the List of Defined Password Validators

- Use `dsconfig` to view the set of password validators defined in the Directory Server.

```
$ bin/dsconfig --no-prompt list-password-validators
```

To Configure the Attribute Value Password Validator

1. Use `dsconfig` to edit the existing default configuration for the Attribute Value Password Validator. For example, the following change configures the validator to only examine a specified set of attributes..

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Attribute Value" \
  --set match-attribute:cn \
  --set match-attribute:sn \
  --set match-attribute:telephonenumber \
  --set match-attribute:uid
```

2. Update an existing password policy to use the Attribute Value Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Attribute Value"
```

3. Test the Attribute Value Password Validator by submitting a password that is identical to one of the configured attributes (`cn`, `sn`, `telephonenumber`, `uid`).

```
$ bin/ldappasswordmodify --authzID "uid=user.0,ou=People,dc=example,dc=com"
 \
  --newPassword user.0
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks
 defined in the
server: The provided password was found in another attribute in the user
 entry
```

To Configure the Character Set Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we change the requirement for special characters making them optional in a password, and add a requirement that at least two digits must be included in the password. Thus, in this example, all newly created passwords must have at least one lowercase letter, one uppercase letter, two digits, and optionally any special characters listed.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Character Set" \
  --remove character-set:1:0123456789 \
  --remove "character-set:1:~\!@#\$\%^&*()-_=+[]{}\|;:,.<>/?" \
  --add character-set:2:0123456789 \
  --add "character-set:0:~\!@#\$\%^&*()-_=+[]{}\|;:,.<>/?" \
  --set allow-unclassified-characters:false
```

2. Update an existing password policy to use the Character Set Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Character Set"
```

3. Test the Character Set Password Validator by submitting a password that meets the requirements (one lowercase letter, one uppercase letter, two digits). The following example should reject the given password because it does not pass the Character Set Password Validator.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abab1
```

To Configure the Length-Based Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the required minimum number of characters in a password to five.

   ```
   $ bin/dsconfig create-password-validator \
     --validator-name "Length-Based Password Validator" \
     --set max-password-length:5 --set min-password-length:5
   ```

2. Update an existing password policy to use the Length-Based Password Validator.

   ```
   $ bin/dsconfig set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Length-Based Password Policy"
   ```

3. Test the Length-Based Password Validator by submitting a password that has fewer than the minimum number of required characters.

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abcd
   ```

   ```
   The LDAP password modify operation failed with result code 53
   Error Message: The provided new password failed the validation checks
    defined in
   the server: The provided password is shorter than the minimum required
    length of
   5 characters
   ```

To Configure the Regular Expression Password Validator

1. Use `dsconfig` to create a Regular Expression password validator. The following password validator checks that the password contains at least one number, one lowercase letter, and one uppercase letter with no restrictions on password length. If the password matches the regular expression, then it will be accepted. When using the following command, remember to include the LDAP/LDAPS connection parameters (host name and port), bind DN, and bind password.

   ```
   $ bin/dsconfig create-password-validator \
     --validator-name "Regular Expression" \
     --type regular-expression --set enabled:true \
     --set "match-pattern:^\\w*(?=\\w*\\d)(?=\\w*[a-z])(?=\\w*[A-Z])\\w*\$" \
     --set match-behavior:require-match
   ```

2. Update an existing password policy to use the Regular Expression validator.

   ```
   $ bin/dsconfig set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Regular Expression"
   ```

3. Test the Regular Expression Validator by submitting a password that meets the requirements (contains one number, one lowercase letter, and one uppercase letter), then run it again with a password that does not meet these requirements.

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaA1
   ```

   ```
   The LDAP password modify operation was successful
   ```

4. Try another password. The following password should fail, because no uppercase letter is present.

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaa1
   ```

   ```
   Error Message: The provided new password failed the validation checks
   ```

```
defined in the server: The provided password is not acceptable because it
 does
not match regular expression pattern '^\w*(?=\w*\d)(?=\w*[a-z])(?=\w*[A-
Z])\w*$'
```

To Configure the Repeated Character Password Validator

1. Use `dsconfig` to edit the existing default configuration.
   - In this example, we set the maximum consecutive length of any character to 3. For example, the following validator rejects any passwords, such as "baaaa1" or "4eeeeb", etc.

     ```
     $ bin/dsconfig set-password-validator-prop \
       --validator-name "Repeated Characters" \
       --set max-consecutive-length:3
     ```
   - Or, you can configure the validator to reject any character from a pre-defined character set that appears more than the specified number of times in a row (2). You can also specify more than one character set. For example, the following validator defines two characters sets: [abc] and [123]. It rejects any passwords with more than two consecutive characters from a character set. Thus, "aaa", "bbb", "ccc", "abc", or "123" and so on fails, but "12a3" is okay.

     ```
     $ bin/dsconfig set-password-validator-prop \
       --validator-name "Repeated Characters" \
       --set character-set:123 --set character-set:abc
     ```
2. Update an existing password policy to use the Repeated Character Password Validator.

   ```
   $ bin/dsconfig --no-prompt set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Repeated Characters"
   ```
3. Test the Repeated Character Validator by submitting a password that has more than the maximum allowable length of consecutive characters.

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" \
     --newPassword baaa1
   ```

   ```
   The LDAP password modify operation failed with result code 53
   Error Message: The provided new password failed the validation checks
    defined
   in the server: The provided password contained too many instances of the
    same
   character appearing consecutively. The maximum number of times the same
   character may appear consecutively in a password is 2
   ```

To Configure the Similarity-Based Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the minimum number of differences to 2.

   ```
   $ bin/dsconfig set-password-validator-prop \
     --validator-name "Similarity-Based Password Validator" \
     --set min-password-difference:2
   ```
2. Update an existing password policy to use the Similarity-Based Password Validator. The `password-change-requires-current-password` property must be set to TRUE, so that the password policy will ensure that the user’s current password is available when that user is choosing a new password.

   ```
   $ bin/dsconfig set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Similarity-Based Password Validator" \
     --set password-change-requires-current-password:true
   ```

3. Test the Similarity-Based Password Validator by submitting a password that has fewer than the minimum number of changes (e.g., 2). The `ldappasswordmodify` command requires the --currentPassword option when testing the Similarity-Based Password Validator.

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" \
     --currentPassword abcde --newPassword abcdd
   ```

   ```
   The LDAP password modify operation failed with result code 49
   ```

To Configure the Unique Characters Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the minimum number of unique characters that a password is allowed to contain to 3.

   ```
   $ bin/dsconfig set-password-validator-prop \
     --validator-name "Similarity-Based" --set min-unique-characters:3
   ```

2. Update an existing password policy to use the Unique Characters Password Validator.

   ```
   $ bin/dsconfig set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Unique Characters"
   ```

3. Test the Unique Characters Password Validator by submitting a password that has fewer than the minimum number of unique characters (e.g., 3).

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" \
     --newPassword aaaaa
   ```

   ```
   The LDAP password modify operation failed with result code 53
   Error Message: The provided new password failed the validation checks
    defined
   in the server: The provided password does not contain enough unique
    characters.
   The minimum number of unique characters that may appear in a user password
    is 3
   ```

Chapter

19

Managing Replication

Topics:

The PingDirectory Server provides a robust replication system to ensure high availability and fast failover in production environments. Write requests can be handled by every server in the topology with the replication component performing immediate synchronization with other members. The replicated server environment ensures that LDAP clients can seamlessly fail over to another server instance.

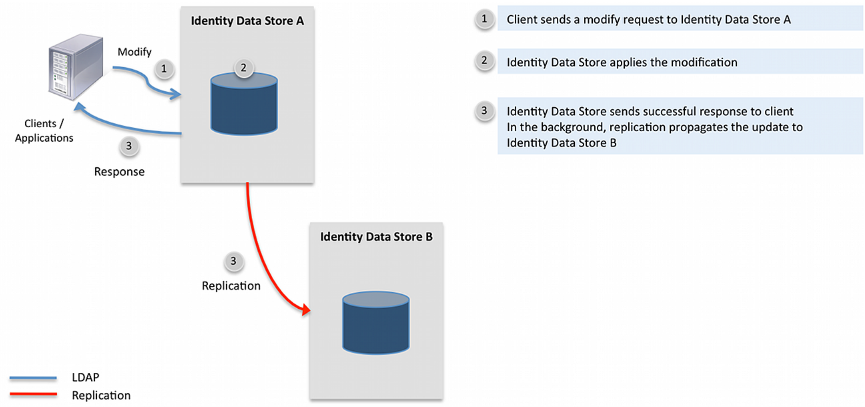
This chapter presents the architectural overview of replication, detailed configuration steps, ways to monitor replication as well as troubleshooting steps:

Overview of Replication

Replication is a data synchronization mechanism that ensures that updates made to a database are automatically replayed to other servers. Replication improves data availability when unforeseen or planned outages occur, and improves search performance by allowing client requests to be distributed across multiple servers.

By default, all Directory Servers participating in replication are writable, so that LDAP clients can perform updates at any of these Directory Server instances. These updates will be propagated to the other servers automatically in the background and applied in the same order as the updates were entered. The replication process flow is designed to immediately propagate changes to the other replicas in the topology with little or no latency.

The following figure demonstrates the basic flow of replication.



1 Client sends a modify request to Identity Data Store A

2 Identity Data Store applies the modification

3 Identity Data Store sends successful response to client
In the background, replication propagates the update to Identity Data Store B

Figure 10: Replication Process Flow

The benefits of replication can be summarized as follows:

- **High-Availability**. Because the data is fully replicated on all other servers in the topology, replication allows participating servers to process all types of client requests. This mitigates any availability issues when a particular server is down due to a planned maintenance or unplanned outage. For those servers that are temporarily unavailable, they will receive updates when they become available again.
- **Improved Search Performance**. Search requests may be directed to any Directory Server participating in replication, which improves search performance over systems that only access single servers. Note, however, that replication does not improve write throughput since updates need to be applied at all servers.
- **WAN Friendly Data Synchronization**. The built-in compression feature in the replication protocol allows efficient propagation of updates over WAN links.

Replication Versus Synchronization

Replication is not a general purpose synchronization facility as it creates replicas with exact copies of the replicated data. Synchronization, on the other hand, can transform data between two different Directory Information Tree (DIT) structures, map attribute types, synchronize subsets of branches and specific object classes. The differences between replication and synchronization are illustrated as follows:

- **Replication cannot Synchronize between Different DIT Structures**. The DN of replicated entries must be the same on all servers. In some situations, it may be desirable to replicate entries with the help of DN mapping that are under different base DNs, but represent the same data, for example `uid=john.doe,ou=people,o=corp` on one server may represent the same user as `uid=john.doe,ou=people,dc=example,dc=com`. This is not supported by replication. Synchronization fully supports this feature.
- **Replication cannot Map Attribute Types or Transform Attribute Values**. In some situations, it may be necessary to map attribute types or transform attribute values when synchronizing data from one server to another. Replication does not support either attribute type mappings or attribute value transformations.

- **Replication does Not Support Fractional Replication**. Replication cannot be configured to replicate a subset of the attribute types from the replicated data set. Synchronization fully supports this feature.
- **Replication does Not Support Sparse Replication**. Replication cannot be configured to replicate entries with a particular object class only. Synchronization fully supports this feature.
- **Replication Requires Full Control of Replicated Data**. When two servers participate in replication, both servers implicitly trust each other using public key cryptography and apply all updates received via replication, which is considered an internal operation. While trust between servers is established between two endpoint servers, synchronization does not require full control of the data. Disparate server system endpoints can be synchronized, such as a PingDirectory Server and a RDBMS database endpoint with each fully in control of its own data.

If replication does not meet your data synchronization requirements, consider using the PingDataSync Server, which provides the versatility and robust performance required for most production environments.

Replication Terminology

The following replication terms are used throughout this chapter.

Table 44: Replication Terminology

Term	Description
Assured Replication	For applications that require immediate access to replicated data or require data consistency over response time, administrators can configure *assured replication*, where data is guaranteed to replicate *before* the response is returned to the client.
Change Number	A 64-bit number used to consistently order replication updates across the entire topology. It is composed of 3 fields: the timestamp of the update (measured in milliseconds since 00:00:00 UTC, January 1, 1970), the Replica ID, and the Sequence Number.
Conflict	Client updates made at different replicas affecting the same entry may be found in conflict when the updates are replayed at another replica. The Change Number of each update allows most of these conflicts to be resolved automatically. Certain updates, such as adding an entry with different attribute values at two servers simultaneously, result in conflicts that are flagged for required manual resolution.
Eventual Consistency	When not using assured replication, recent updates from LDAP clients are not immediately present at all servers. Out-of-sync data will be eventually synchronized and will be consistent on all servers. The network latency typically controls how long a given update takes to replicate.
Global Administrator	The administrative user with full rights to manage the replication topology. The user is created at the time of replication enable between two non-replicating servers and thereafter copied to newly enabled servers. The replication command-line utility expects the user name of the Global Administrator (by default, admin). The user is stored in `cn=Topology Admin Users,cn=Topology,cn=config`.
Historical Data	Historical Data are records of attribute value changes as a result of an LDAP Modify Operation. Historical Data is stored in the `ds-sync-hist` attribute of the target entry. This information allows replication to resolve conflicts from simultaneous LDAP Modify operations automatically.
Location	The collection of servers that may have similar performance characteristics when accessed from this Directory Server or reside in the same data center or city. A separate location setting may be defined for each data center, such as Austin, London, Chicago, etc. Location settings are used in the selection of the WAN Gateway server.

Term	Description
Modify Conflict	A conflict between two LDAP Modify operations. Conflicts arising from LDAP Modify operations are automatically resolved.
Naming Conflict	Any conflict other than Modify Conflicts. Naming conflicts typically include an operation that changes the DN of the target entry, creates an entry, or deletes an entry at one Replica.
Replica	The component in the Directory Server that handles interaction with a single replication domain, for example, the `dc=example,dc=com` replication domain within the `userRoot` backend.
Replica ID	The unique identifier of a replica that is set automatically in the corresponding Replication Domain configuration entry at each participating server. The replica ID identifies the source of each update in the replication topology.
Replica State	A list of the most recent change numbers of replication updates that have been applied to a replica. There can be at most one change number from each replica in the state. The replica state helps the Replication Server component to determine which updates the Replica has not received yet.
Replication	An automated background process that pushes directory server data changes to all other replicas.
Replication Changelog	A backend maintained by each replication server independently that records updates from each replica. This backend is distinct from the LDAP Changelog and the two should not be confused. The main distinction is as follows: • The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at `<server-root>/db/changelog`. • The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at `<server-root>/changelogDB` and is not accessible by clients and server extensions.
Replication Domain	The data configured for replication as defined by the base DN. Updates to entries at and below the base DN will be replicated.
Replication Replay	When a replica locally applies the update received via a replication server.
Replication Server	A component within the Directory Server process that is responsible for propagating directory server data changes to and from replicas. Each Directory Server instance participating in replication is also running a replication server.
Replication Server ID	The unique identifier of a replication server set automatically in its configuration object at each server in the replication topology. This identifier is used in the connection management code of the replication servers.
Replication Server State	A list of the most recent change numbers of replication updates that have been processed by a replication server. There can be at most one change number from each replica in the topology. The Replication Server State is used to determine which updates need to be sent to other replication servers. Similarly, the replica can use the Replication Server State to identify the set of updates to send to the replication server.
Sequence Number	A field in the change number that indicates the sequence of the client updates at a particular replica. For every update at the replica, the number is incremented. The initial value of the sequence number is 1. The number is stored as a 32-bit integer, but only positive values are used. The sequence number can roll over.
WAN Gateway	The designated replication server that assumes the WAN Gateway role within a collection of co-located replication servers (i.e., servers that are defined with identical location settings). Replication update messages between servers at different locations are routed through WAN gateways. The WAN Gateway role is assigned automatically

Term	Description
	by the protocol based on the server's WAN Gateway Priority setting. If the WAN Gateway server is down for any reason, the server with the next highest WAN Gateway Priority will dynamically assume the WAN Gateway role.
WAN Gateway Priority	The configuration setting that determines which replication server assumes the WAN Gateway role. The replication server with the lowest WAN Gateway Priority value in a location assumes the role of the WAN Gateway. The priority values can be set to 0 (never be a gateway), or any value from 1 (highest priority) to 10 (lowest priority).

Replication Architecture

The major elements of replication in the PingDirectory Server are introduced in this section.

Eventual Consistency

Replication is based on an eventual-consistency model, where updates that are propagated through a connected network of servers will eventually be consistent on all servers over a very short period of time. In a typical update operation, a client application updates an entry or group of entries on the PingDirectory Server with an ADD, DELETE, MODIFY, or MODIFY DN operation. After processing the operation, the Directory Server returns an LDAP response, while concurrently propagating the update to the other servers in the replicated topology. This concurrent processing model allows the client to continue submitting update requests without waiting for a replication completion response from the server. Alternatively, assured replication can be configured for specific write requests that requires local or global consistency, across datacenter locations, before a response is returned to the client. For more information, see *Configuring Assured Replication* on page 340.

To support this processing model, replication never locks the targeted entries at the other Directory Server instances before an update can be made locally. This means that the replicated Directory Servers may have an inconsistent view of the targeted entry for a very short period of time but will catch up as the propagated changes are applied. The eventual-consistency model also allows clients to complete update operations faster, since clients do not have to wait for replication to propagate the change. The rate of update operations remains the same no matter how many Directory Servers participate in replication.

Replicas and Replication Servers

Each Directory Server has an embedded replication server that is responsible for transmitting updates to other replication servers. There is a separate component, called a replica, for each replicating base DN, such as `cn=schema, dc=example,dc=com`. Each replica connects to the embedded replication server running within the Directory Server JVM process.

When a client application updates an entry on the Directory Server, the replica sends an update message to its embedded replication server. The replication server applies the change to the `replicationChangelog` backend repository and then sends an update message to the connected replication server on another directory server. The replication server on other directory servers then passes the change to the appropriate replica, which in turn applies the change to its backend after performing conflict resolution. This standard setup is seen in the figure below.

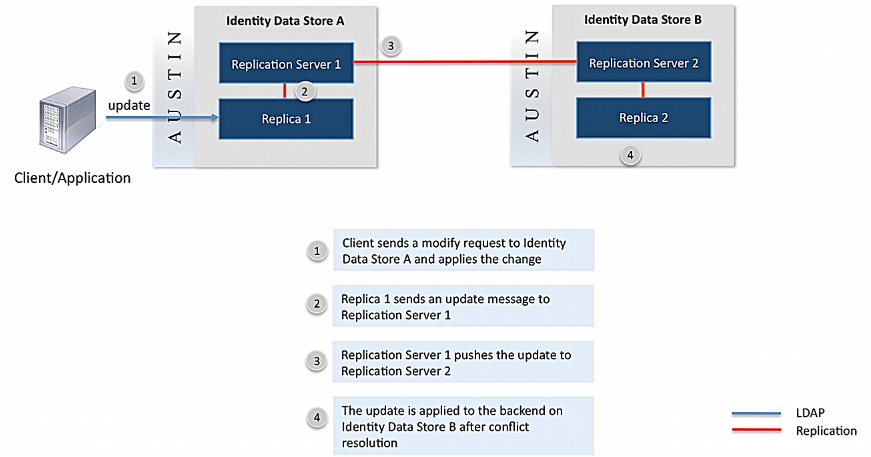

Identity Data Store A
Identity Data Store B
AUSTIN
AUSTIN
Replication Server 1
Replica 1
Replication Server 2
Replica 2
update
Client/Application
1
2
3
4
Client sends a modify request to Identity Data Store A and applies the change
Replica 1 sends an update message to Replication Server 1
Replication Server 1 pushes the update to Replication Server 2
The update is applied to the backend on Identity Data Store B after conflict resolution
LDAP
Replication

Figure 11: Replicas and Replication Servers

Authentication and Authorization

The authentication in the Replication Protocol is based on public key cryptography using client certificate authentication via TLS. The certificate used for authentication is stored in the `ads-truststore` backend of the Directory Server. During replication setup, the command-line utility distributes public keys to all directory servers to establish trust between the Directory Servers and to enable client authentication via TLS.

The authorization model of replication is simple: once authenticated, the remote Directory Server is fully authorized to exchange replication messages with the local Directory Server. There is no other access control in place.

Logging

The access log messages in the Directory Server indicate if the update was received via replication and includes the corresponding change number. This allows the administrator to track which Directory Server the update originated from.

Replication Deployment Planning

The following should be considered before deploying in a production environment:

- **Minimizing Replication Conflicts**. Attention should be paid to the origin of client write requests to prevent a conflict. If two different clients attempt to create an entry with the same name, DN, at the same time against different servers, the possibility exists that both client requests will succeed. In this case, a conflict alert will be sent by the server. However, understand the client traffic pattern beforehand will minimize these occurrences. The Directory Server's Assured Replication feature and the PingDirectoryProxy Server can both assist in minimizing conflicts.
- **Replication Purge Delay**. Adjust the default one-day replication purge delay, consistently across all servers, to accommodate automatic catchup of changes when a server is offline for an extended period of time. The replication changes database, stored in `<server-root>/changelogDb`, grows larger as the replication purge delay is increased. A minimum value should be defined.

The rest of this section highlights other topics of consideration.

Location

In multi-site deployments, it is strongly recommended to configure the directory servers with location information using the `dsconfig create-location` command and `dsconfig set-global-configuration-prop` command. The Directory Server cannot determine LAN boundaries automatically, so incorrect location settings can result in undesired WAN communication. By default, replication also compresses all traffic between directory servers in different locations.

We recommend setting up the locations prior to enabling replication. The `dsreplication enable` command prompts for location information if you have forgotten to define the property.

User-Defined LDAP

Directory Servers participating in replication are required to have a uniform user-defined schema. The `dsreplication` command-line utility sets up replication for the schema backend the first time replication is enabled to ensure that future schema changes are propagated to all directory servers.

Disk Space

Replication increases the disk space required for the Directory Server. The Replication Changelog backend keeps changes from all directory servers for 24 hours by default. After this time period, also known as the purge delay, the backend is trimmed automatically.

In addition, within the `userRoot` and other local DN backends, attribute-level changes are recorded for a short period of time in the `ds-sync-hist` attribute of the entry targeted by an LDAP Modify operation. This attribute is used to resolve all conflicts resulting from LDAP Modify operations automatically.

The disk space impact of replication is highly dependent on the rate and size of changes in the replication topology, and the Replication Changelog purge delay.

Memory

Compared to a standalone directory server, replicated directory server instances require slightly more memory. All of the items discussed in the Disk Space section have an impact on the amount of memory the Directory Server is using. The additional replication overhead is typically less than 5%.

Time Synchronization

Even though replication has a built-in mechanism to compensate for the potential clock skew between hosts, it is generally recommended to keep system clocks in sync within the deployment.

Communication Ports

The replication server component in each directory server listens on a TCP/IP port for replication communication (the replication server port). This port, typically 8989, must be accessible from all directory servers participating in replication. The server-to-server communication channel is kept alive using a heartbeat, which occurs every 10 seconds. This traffic will prevent firewalls from closing connections prematurely.

The replication command-line utility (`dsreplication`) requires access to all directory servers participating in replication. This includes the LDAP or LDAPS port of the directory servers.

Keep these communication requirements in mind when configuring firewalls.

Hardware Load Balancers

Replication allows writes to be directed to any directory server in the topology. Distributing write operations in a round-robin fashion, however, may introduce conflicts. In particular, distributing a series of LDAP Add, Delete and Modify DN operations targeting the same DN in quick succession can result in conflicts that require manual intervention. The Assured Replication feature can help prevent conflicts created by the same client application.

If possible, consider using server affinity with the load balancer that either associates a client IP address or a client connection with a single directory server instance at a time. Also, consider using the PingDirectoryProxy Server for load balancing LDAP traffic. The Server Affinity feature in the Directory Proxy Server enables replication-friendly load balancing.

Directory Proxy Server

In addition, to facilitate replication-friendly load balancing, the PingDirectoryProxy Server should be considered in every replication deployment. The Directory Proxy Server can automatically adapt to conditions in backend directory servers using health checks and route traffic accordingly. For example, traffic can be re-routed from directory servers with large backlog of replication updates.

To Display the Server Information for a Replication Deployment

- Run the `dsreplication status` command with the --displayServerTable option.

```
$ bin/dsreplication status --displayServerTable
```

```
          --- Replication Status for dc=example,dc=com: ---
Server                          : Location : Priority (1) : Status
--------------------------------:----------:--------------:----------
austin-01.example.com:8989      : US       : 1 (*)        : Available
austin-02.example.com:8989      : US       : 2            : Available
london-01.example.com:8989      : UK       : 5            : Available
london-02.example.com:8989      : UK       : 5 (*)        : Available
sydney-01.example.com:8989      : AU       : 5 (*)        : Available
sydney-02.example.com:8989      : AU       : 5            : Available

[1] WAN Gateway Priority. WAN gateways are marked with a *. To minimize
WAN utilization, the server with the WAN Gateway role is the only server
in a location that exchanges updates with remote locations.

--- Replication Status for dc=example,dc=com: Enabled ---
Server                          : Location : Entries : Conflict Entries :
 Backlog : Recent Change Rate
--------------------------------:----------:---------:------------------:---------:---
austin-01.example.com:1389      : US       : 3478174 : 0                : 8
      : 333
austin-02.example.com:1389      : US       : 3478174 : 0                : 5
      : 345
london-01.example.com:1389      : UK       : 3478174 : 0                : 0
      : 349
london-02.example.com:1389      : UK       : 3478174 : 0                : 0
      : 5
sydney-01.example.com:1389      : AU       : 3478173 : 0                : 0
      : 350
sydney-02.example.com:1389      : AU       : 3478270 : 0                : 30
      : 332
```

To Display All Status Information for a Replication Deployment

- Run the `dsreplication status` command with the --showAll option. You can also use the --showAll option together with the --displayServerTable option to see the server table information for the replication topology.

Enabling Replication

Enabling replication between multiple Directory Servers means that any change within the replicated base DN is automatically propagated to all other Directory Servers in the topology. Configuration changes are not replicated, but can be through the use of Server Groups. Therefore, each Directory Server should be configured according to the deployment design.

Overview

To interface with the replication topology, the Directory Server provides a command-line utility, `dsreplication`, that must be used to manage and monitor replication.

Replication setup involves the following basic steps:

- **Set up the servers**. This is the basic installation steps to set up a Directory Server instance.

- **Import or restore data to one server**. After setting up the servers, at least one server should have the target data loaded through `import-ldif` or restore.
- **Enable replication between the servers**. Using the `dsreplication` tool, enable replication for each server to be included in the replication topology. The `dsreplication enable` subcommand should be run *N - 1* times for a topology of *N* servers. See *Command Line Interface* for more information.
- **Initialize data from source server to every server in the topology**. Run the `dsreplication initialize` subcommand for every target server that needs a copy of the data from the source server.
- **Verifying the replication topology**. Administrators can check the replication status after configuring the topology using the `dsreplication` status tool.

Command Line Interface

Replication topologies are configured and maintained using the `dsreplication` command-line utility, which supports interactive and non-interactive modes. If you are running the server for the first time, we recommend using the `dsreplication` tool in interactive mode.

The `dsreplication` tool has the following format including some important subcommands listed in the section *The dsreplication Command-Line Utility*:

`dsreplication` {subcommand} {connection parameters}

The `dsreplication` tool keeps a history of invocations in the `logs/tools/dsreplication.history` file and keeps a log of up to 10 `dsreplication` sessions in the `logs/tools` directory.

What Happens When You Enable Replication

The `dsreplication` enable subcommand is used to set up replication. The enable subcommand carries out the following functions:

- If it does not already exist, the global administrator user is created. The global administrator user has all the rights and privileges to update replication-related configuration objects. Most `dsreplication` subcommands require the global administrator.
- The server instances are registered in the `cn=topology,cn=config` tree. The registration includes basic host name, port information as well as the public key used during the replication authentication process.

  In case both servers are already participating in replication, the `cn=topology,cn=config` is merged to retain the server information from existing topologies.
- The embedded replication server is enabled. Servers already in replication will see their replication server configuration updated with the information of the new replication server.
- A replication domain is created for the requested base DNs. In case the first base DN is enabled, the replication domains for two additional base DNs are also enabled: `cn=topology,cn=config` and `cn=schema`.
- Initialization for the `cn=topology,cn=config` base DN is executed. This will ensure that `cn=topology,cn=config` is uniform across the replication topology.
- Initialization for the `cn=schema` base DN is executed. This will ensure that a uniform schema is present in the replication topology.
- Initialization must be performed for the enabled base DNs.

Initialization

Replica initialization transfers of a copy of the backend containing the replication domain to a target server. This should be performed after replication is enabled with the `dsreplication initialize` subcommand. There is no impact on the source server during this process.

Note: When enabling or initializing servers, the `--topologyFilePath` option can be used with `dsreplication` to specify a file with a series of hosts and ports available in the topology that can be used as source servers. This option is used in place of specifying host 1, port1. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available for replication. This file is generated with the `manage-topology export` command.

- **dsreplication initialize**. The recommended approach for replica initialization. The `dsreplication initialize` subcommand performs the most efficient copy of data needed to initialize one or more replicas on a target server. Any existing data on the target server replica will be lost and the backend containing the base DN will be taken offline on the target server during the initialization.
- **Binary Copy**. The database copy method involves copying database backup files from the source directory server to one or more target servers. The Directory Server provides tools necessary for backing up and restoring backends. Using `server-root/bin/backup`, create a backup of the backend containing the replicated base DN. The backup files then need to be transferred to the target server(s) and restored individually with `server-root/bin/restore`. There are additional considerations when using database copy as the means to initialize a target replica:

  - If encryption is enabled on the servers, then a database `bin/encryption-settings export` then `bin/encryption-settings import`must be performed on the `encryption-settings` backend.
  - When using database copy to initialize a server which has been offline longer than the replication purge delay, the database copy of the `replicationChanges`, `schema`, and `adminRoot` backends are required.

Replica Generation ID

Each replica has a generation ID, which is an integer that summarizes the replica. It provides replication with a quick and simple means of determining if two replicas contain the same data. If they do contain the same data, they'll have the same generation ID.

When replication is operating correctly, all of the replicas for each replicated base DN will have the same generation ID. The generation ID is stored on each replica as the operational attribute `ds-sync-generation-id`. For example:

```
ldapsearch -b 'dc=example,dc=com' -s base '(&)' ds-sync-generation-id
 dn: dc=example,dc=com
 ds-sync-generation-id: 2058329333
```

When the server starts, or when replication is enabled, the generation ID is computed for each affected replica that does not already have a generation ID stored as `ds-sync-generation-id`. The following is used to calculate the generation ID:

- The total number of entries in the replica. This is referred to as "the count."
- The first 1000 entries in the replica are converted to normalized LDIF, which is referred to as "the LDIF." Normalized LDIF only includes attributes `objectclass`, `sn`, `cn` and `ds-entry-unique-id`, and uses OIDs in place of attribute names.
- The Adler-32 checksum is calculated with the string produced by concatenating the count and the LDIF as input. This Adler-32 checksum is the generation ID.
- The generation ID is stored on the base DN as `ds-sync-generation-id`. This is so that the ID does not need to be computed the next time the replica is loaded.

Deploying a Basic Replication Topology

This section describes how to set up a two-server replication topology. The example uses the LDAP and replication server ports 1389 and 8989 respectively.

Table 45: Replica Ports

Host Name	LDAP Port	Replication Port
server1.example.com	1389	8989
server2.example.com	1389	8989

To Deploy a Basic Replication Deployment

1. Install the first directory server with 2000 sample entries.

```
$ ./setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389
 \ --rootUserPassword pass --sampleData 10000 --no-prompt
```

2. Install the second directory server either on a separate host or the same host as the first, but with a different LDAP port.

```
$ ./setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389
 \ --rootUserPassword pass --no-prompt
```

3. From the first server, run the `bin/dsreplication` command in interactive mode to configure a replication topology:

```
$ bin/dsreplication
```

4. From the Replication Main menu, select the Manage the topology option.

```
>>>> Replication Main Menu

What do you want to do?

    1) Display replication status
    2) Manage the topology (add and remove servers)
    3) Initialize replica data over the network
    4) Initialize replica data manually
    5) Replace existing data on all servers

    q) quit

Enter choice: 2
```

5. From the Manage Replication Topology menu, choose the Enable Replication option.

```
>>>> Manage Replication Topology

Select an operation for more information.

    1)  Enable Replication -- add or re-attach a server to the topology
    2)  Disable Replication -- permanently remove a running replica from the
 topology
    3)  Remove Defunct Server -- permanently remove an unavailable server
 from the
         topology
    4)  Cleanup Server -- remove replication artifacts from an offline,
 local server
         (allowing it to be re-added to a topology)

    b)  back
    q)  quit

Enter choice [b]: 1
```

6. On the Enable Replication menu, read the brief introduction on what will take place during the setup, and then, enter "c" to continue the enable process.
7. Next, enter the LDAP connection parameters for the first of the two replicas that you are configuring. First, enter the host name or IP address of the first server.
8. Next, enter the type of LDAP connection to the first server: 1) LDAP, 2) LDAP with SSL, or 3) LDAP with StartTLS.
9. Type the LDAP listener port for the first replica. If you are a root user, you will see port 389 as the default. Others will see port 1389.

10. Authenticate as a root DN, such as `cn=Directory Manager`. You will be prompted later in the process to set up a global administrator and password. The global administrator is the user ID that manages the replication topology group.
11. Repeat steps 7–10 for the second replica.
12. Next, the `dsreplication` tool checks for the base DN on both servers. In order to enable replication, data must be present on at least one of the servers. For this example, press Enter to select the default base DN, `dc=example,dc=com`.

    ```
    Choose one or more available base DNs for which to enable replication:

        1)  dc=example,dc=com

        c)  cancel

    Enter one or more choices separated by commas [1]:
    ```

    Note: If you see the following message:

    ```
    There are no base DNs available to enable replication between the two
     servers.
    ```

    In most cases, a base DN was not set up on one of the directory servers or the backend is disabled.
13. Next, the prompt asks if you want to set up entry balancing using the Directory Proxy Server. Press Enter to accept the default (no), since we are not setting up replication in an entry-balanced environment in this scenario. For more information, see the *PingDirectoryProxy Server Administration Guide*.

    ```
    Do you plan to configure entry balancing using the Directory Proxy Server?
     (yes / no) [no]:
    ```
14. Next, enter the replication port for the first replica (default, 8989). The port must be free.
15. If the first server did not have a pre-defined location setting, `dsreplication` will prompt you to enter a location. Press Enter to accept the default (yes) to set up a Location for the first server. Enter the name of the server's location.

    ```
    The first server has not been configured with a location.
    Assigning a location to each server in the replication topology reduces
    network traffic in multi-site deployments. Would you like to set the
    location in the first server? (yes / no) [yes]

    The location of the first server: Austin
    ```
16. Repeat the previous steps for the second directory server. Again, if you did not pre-define a location setting for the second server, you will be prompted to enter this information.
17. At this time, set up the Global Administrator user ID (default is "admin") and a password for this account. The Global Administrator user ID manages the directory servers used in the replication topology.

    ```
    Specify the user ID of the global administrator account
    that will be used to manage the Ping Identity Directory Server
    instances to be replicated [admin]:

    Password for the global administrator:
    Confirm Password:
    ```
18. Return to the Replication Main Menu and enter the number corresponding to initializing data over the network.
19. On the Initialize Replica Data over the Network menu, select Initialize to initialize data on a single server, and then enter `c` to continue.
20. Next, specify a server in the replication topology. For this example, enter the host name or IP address, LDAP connection type, LDAP port, Global Admin user ID and password of the first server.
21. Next, select the source server that is hosting the data to which the target server will be initialized. For this example, select the first server, since the sample dataset has been loaded onto this server.

22. Next, select the base DN that will be initialized. In most cases, the base DN for the root suffix will be replicated. In this example, `dc=example,dc=com`.
23. Next, select the second server in this example that will have its data initialized, and then enter the Global Admin user ID and password for the target server. Any data present on the target server will be over-written.
24. Press Enter to confirm that you want to initialize data on the target server. When completed, you should see "Base DN initialized successfully."

```
Initializing the contents of a base DN removes all the existing contents of
that base DN. Do you want to remove the contents of the selected base DNs on
 server
server2.example.com:1389 and replace them with the contents of server
server1.example.com:1389? (yes / no) [yes]:
```

25. On the Initialize Replica Data Over Network menu, enter `b` to back out one level to the main menu. Then, on the Replication Main menu, enter the number to view the replication status.

```
     --- Replication Servers: dc=example,dc=com ---
Server                   : Location : Conflict Entries : Backlog : Recent
 Change Rate
-------------------------:----------:------------------:---------:--------------------
ds1 (example.com:1389)   : austin   : 0                : 0       : 0
ds2 (example.com:1389)   : austin   : 0                : 0       : 0
```

A Deployment with Non-Interactive dsreplication

This example will create a four-server topology spanning two data centers. The four servers are already installed and have locations Austin and Budepest defined.

Note: When enabling or initializing servers, the `--topologyFilePath` option can be used with `dsreplication` to specify a file with a series of hosts and ports available in the topology that can be used as source servers. This option is used in place of specifying host 1, port1. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available for replication. This file is generated with the `manage-topology export` command.

To Deploy with Non-Interactive dsreplication

1. Generate the sample data using the `make-ldif` tool on the first server.

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-10K.template \
  --ldifFile ldif/10K.ldif
```

2. Select a server from which to import data, and to be the source for future initialization to other servers. Stop this server, import the sample LDIF and start again, or perform a task-based `import-ldif` with the connection options.

```
$ bin/stop-server
$ bin/import-ldif --backendID userRoot --ldifFile ldif/10K.ldif
$ bin/start-server
```

3. Enable replication by choosing a specific server and running `dsreplication enable` three times. For the first invocation, create the replication topology administrator with the name `admin`.

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 austin02.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt
```

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 budapest01.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt

$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 budapest02.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt
```

4. Initialize the other servers (the dc=example,dc=com replicas) from the server that had the data imported with `import-ldif`. To minimize the WAN transfers, initialize `budapest02` from `budapest01`.

```
$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
  --hostDestination austin02.example.com --portDestination 1389 \
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt

$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
  --hostDestination budapest01.example.com --portDestination 1389 \
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt

$ bin/dsreplication initialize --hostSource budapest01.example.com --
portSource 1389 \
  --hostDestination budapest02.example.com --portDestination 1389 \
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt
```

5. View the state of replication.

```
$ bin/dsreplication status --adminPassword password --no-prompt --
displayServerTable --showAll
```

To Use dsreplication with SASL GSSAPI (Kerberos)

This example procedure assumes that you have configured SASL GSSAPI on all servers in the replication topology and that they are working properly.

The Directory Server's utilities all support SASL GSSAPI options for systems using Kerberos as its main authentication mechanism. The following procedure shows how to use `dsreplication` with SASL GSSAP to set up a new `replication.admin` identity, while enabling replication on a server. The following are important points about the configuration:

- A separate Kerberos identity is required to manage replication. Existing Kerberos credentials can be used to interact with the server when enabling replication and creating the new identity.
- The new identity, such as replication.admin, must not exist as the cn or uid value under any public base DN.

1. Set the LDAP Connection Handler to explicitly listen on the server's hostname address.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --remove listen-address:0.0.0.0 --add listen-address:host.example.com
```

2. Update the identity mapper to have `cn=topology,cn=config` included in the list of base DNs and to add the `cn` attribute to match attributes. This step is required to map the admin account to the Kerberos realm.

```
$ bin/dsconfig set-identity-mapper-prop \
  --mapper-name "Regular Expression" \
  --add match-attribute:cn \
  --set "match-base-dn:cn=topology,cn=config" \
  --set match-base-dn:dc=example,dc=com
```

3. Invoke replication enable, authenticating as the existing kerberos `authid`. Note that no bind DNs and passwords are required to authenticate because we are using SASL binding. However, the new replication admin user requires a password at creation time, so we recommend using a strong random password. Once SASL is working, you will no longer have to provide this random password. Also note that we are forcing the use of the ticket cache, so make sure you have properly authenticated as `ds.admin` from your local host and the ticket is not expired in the cache.

```
$ kinit -p ds.admin
$ bin/dsreplication enable \
  --host1 server1.example.com --port1 1389 --replicationPort1 1989 \
  --host2 server2.example.com --port2 2389 --replicationPort2 2989 \
  --baseDN dc=example,dc=com \
  --adminUID replication.admin --adminPassword strongPassword \
  --saslOption1 mech=gssapi --saslOption1 authid=ds.admin@EXAMPLE.COM \
  --saslOption1 useTicketCache=true --saslOption1 requireCache=true \
  --saslOption2 mech=gssapi --saslOption2 authid=ds.admin@EXAMPLE.COM \
  --saslOption2 useTicketCache=true --saslOption2 requireCache=true
```

4. Use `dsreplication initialize` to initialize data on remote server.

```
$ kinit -p replication.admin
$ bin/dsreplication initialize \
  --hostSource server1.example.com --portSource 1389 \
  --hostDestination server2.example.com --portDestination 2389 \
  --baseDN dc=example,dc=com \
  --saslOption mech=GSSAPI \
  --saslOption authID=replication.admin@EXAMPLE.COM  \
  --no-prompt
```

5. The temporary `userPassword` can now be deleted from the `replication.admin` entry. Create a file called `remove-password.ldif` with these contents:

```
dn: cn=replication.admin,cn=Administrators,cn=topology,cn=config
changetype: modify
delete: userPassword

```

6. Apply the modifications using `ldapmodify`:

```
$ ./ldapmodify --filename remove-password.ldif -o mech=GSSAPI
-o authid=replication.admin@example.com \
  --saslOption useTicketCache=true \
  --hostname host.example.com --port 1389 \
  --noPropertiesFile
```

7. Check the topology's status by running `dsreplication status`. The `--saslOption useTicketCache=true` and `--saslOption requireCache=true` properties, instead of providing a password, for all `dsreplication` commands after properly creating the admin accounts and mappers.

```
$ bin/dsreplication status \
  --saslOption mech=gssapi \
  --saslOption authid=replication.admin@EXAMPLE.COM \
  --saslOption useTicketCache=true --saslOption requireCache=true \
  --hostname host.example.com --port 1389 \
  --no-prompt
```

Configuring Assured Replication

The PingDirectory Server's replication mechanism is based on the eventual-consistency model, which is a loosely-connected topology that propagates updates to other servers without waiting for their corresponding replication response messages. As a result, there is a small window of time where updates are not all present on the replicas as the changes are replicating through the system. There are, however, deployments that require *immediate* access to replicated data. In such cases, administrators can configure *Assured Replication*, which ensures that replication has completed *before* the update response is returned to the client.

From the LDAP client's perspective, assured replication has no bearing on the result code of the operation, just on the time in which it takes to receive the response for those requests in which replication assurance is applied. Detailed information regarding assurance processing is available to an LDAP client with awareness of the assured replication control.

The assured replication mechanism takes advantage of server location to distinguish between local and remote servers to allow different policies to apply. For example, a common assurance approach is to respond to a client update after all servers in the same location have applied the update, and one or more servers in remote locations have received the update. In addition, the level of assurance applied to each operation can be explicitly requested by the client and/or specified by the server configuration using the Replication Assurance Policy.

Assured replication is supported by client requests directly to PingDirectory Server and/or through a PingDirectoryProxy Server.

About the Replication Assurance Policy

Assured replication uses a *Replication Assurance Policy* to define the properties needed to ensure that replication has satisfactorily completed before the update response is returned to the client. Multiple policies can be defined but only one policy is matched with a client update request. Each policy contains an evaluation order index which, together with an optional request and connection criteria, provides flexibility in matching a policy to request.

The Replication Assurance Policy defines local and remote assurance *levels*. A *level* defines how rigorous the policy should be when waiting for propagation of updates, while *local* and *remote* distinguish servers in the same location versus servers in remote locations. Although optional, it is recommended that request or connection criteria be associated with a policy to apply replication assurance appropriately.

The Directory Server contains a Default Replication Assurance Policy that is enabled but has no assurance levels assigned. In addition to using the Default Replication Assurance Policy, any number of policies can be created and enabled. When a client request is received, the server iterates through the list of enabled policies according to each policy's *evaluation-order-index* property. A smaller *evaluation-order-index* value (e.g., 1) has precedence over policies with larger *evaluation-order-index* values (e.g., 2, 3, 4, etc.). The *evaluation-order-index* values do not need to be contiguous. The first policy that matches a request is associated with the operation.

The Replication Assurance Policy, which is defined on the PingDirectory Server and not on the PingDirectoryProxy Server, has the following properties:

- **evaluation-order-index**. Determines the evaluation order (the smaller value has precedence) among multiple Replication Assurance Policies that match against an operation.
- **local-level**. Specifies the assurance level used to replicate to local servers. A local server is defined as a server with the same `location` property value as set in the global configuration. The `local-level` property must be set to an assurance level as strict as the `remote-level` property. For example, if the `remote-level` is set to "`processed-all-remote-locations`," then the `local-level` property must be "`processed-all-servers`."

    None. Replication to any local server is not assured.

    received-any-server. At least one available local server must receive a replication update before a response is sent to the client.

    processed-all-servers. All available local servers must complete replay of the replication update before the response is sent to the client. If a singular server is enabled, or only one server is available for a particular location, `processed-all-servers` will return a value of `false`.

- **remote-level**. Specifies the assurance level used to replicate to remote servers. A remote server is defined as a server that has a different `location` property value as set in the global configuration.

  None. Replication to any remote server is not assured.

  received-any-remote-location. At least one server at any available remote location must receive a replication update before a response is sent to the client.

  received-all-remote-locations. All available remote servers must receive a replication update before the response is sent to the client.

  processed-all-remote-servers. All available servers at all locations must complete replay of the replication update before the response is sent to the client. If a single server is enabled, or only one server is available for a particular location, `processed-all-remote-servers` will return a value of `false`.
- **timeout**. Specifies the maximum length of time to wait for the replication assurance requirements to be met before timeout and replying to the client.
- **connection-criteria**. Specifies connection criteria used to indicate which operations from clients matching this criteria use this policy. If both connection criteria and request criteria are specified for a policy, then both must match an operation for the policy to be assigned.
- **request-criteria**. Specifies request criteria used to indicate which operations from clients matching this criteria use this policy. If both connection criteria and a request criteria are specified for a policy, then both must match an operation for the policy to be assigned.

Servers in a replication topology are not required to share a homogeneous set of policies; you can configure the Replication Assurance Policies differently on the replicas in a topology. For example, if you configure server A to match add operations to a `processed-all-servers` assurance level, and server B to match add operations to a local `received-any-server` level, then add operations received on server A will have the assurance level of `processed-all-servers` and add operations received on server B will have the assurance level of `received-any-server`.

For more detailed information, see the *PingDirectory Server Configuration Reference Guide*.

Points about Assured Replication

The following are some points when implementing Assured Replication:

- **Client Controls**. The client may optionally include an assured replication request control with each operation. This control allows the client to specify bounds on assurance levels and/or override the timeout assigned by the matching replication assurance policy for the associated operation. The server always honors these request controls. See *About the Assured Replication Controls* for more information.
- **Directory Proxy Server**. Replication assurance policies are not supported on the PingDirectoryProxy Server. Replication client controls are passed through to the underlying Directory Server backend.
- **Schema backend Replication**. The schema backend is not supported by Assured Replication. Replication assurance policies that include criteria to match this backend will be rejected.
- **Backward Compatibility**. Server versions that support assured replication are backwards-compatible with prior versions that do not support assured replication.
- **WAN-Friendly Replication**. Assured replication functions independently from WAN-Friendly Replication and the notion of WAN Gateways.
- **Global Configuration Properties**. The Directory Server provides two configurable global configuration properties that determine the timing of the assurance source and maximum number of replication backup updates to be recognized as an available source.

  - **replication-assurance-source-timeout-suspend-duration**. Specifies the amount of time a replication assurance source will be suspended from assurance requirements if it experiences an assurance timeout. While suspended, the source will be excluded from assurance requirements for all operations originating on this Directory Server. This avoids the situation of repeated timeouts caused by degraded or offline servers. Default is 10 seconds.
  - **replication-assurance-source-backlog-fast-start-threshold**. Specifies the maximum number of replication backlog updates a replication assurance source can have and be immediately recognized as an available source. If a source connects to this server with more than the configured threshold backlog updates, it will be excluded

from assurance requirements for all operations originating from this Directory Server until it completes at least one assurance successfully (i.e. this Directory Server receives an update acknowledgement message from it within the timeout window). Default is 1000.

To Configure Assured Replication

This example illustrates configuring a variety of assured replication policies. In practice it's common for all servers to have the same policy. The following example assumes that three servers are configured on localhost, on ports 1389, 2389 and 3389. Note that each server has a default Replication Assurance Policy with no assurance levels set.

1. On server 1, use `dsconfig` to create request criteria for add operations. This request criteria will be used to match any add operation with the Replication Assurance Policy that will be configured in the next step.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Adds \
  --type simple \
  --set operation-type:add
```

2. On server 1, set up the Replication Assurance Policy to make all add operations assured with a level of `processed-all-servers`, which indicates that all local servers in the topology must complete replay of the replication update before the response is sent to the client. Specify the Adds request criteria configured in the previous step.

```
$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Adds Processed All Locally" \
  --set evaluation-order-index:1 \
  --set local-level:processed-all-servers \
  --set "timeout:500ms" \
  --set request-criteria:Adds
```

3. On server 1, repeat the previous two steps for modify operations. The Replication Assurance Policy "Mods Received Any Locally" ensures that at least one available local server must receive a replication modify update before a response is sent to the client.

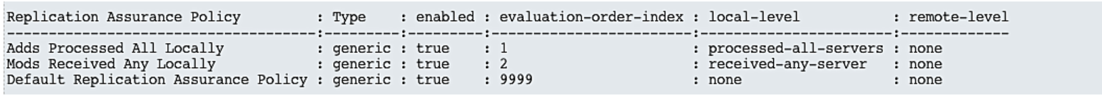
```
$ bin/dsconfig create-request-criteria \
  --criteria-name Mods \
  --type simple \
  --set operation-type:modify

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Mods Received Any Locally" \
  --set evaluation-order-index:2 \
  --set local-level:received-any-server \
  --set "timeout:500ms" \
  --set request-criteria:Mods
```

4. On server 2, repeat steps 1-3 to set up the Adds and Mods request criteria and its respective Replication Assurance Policy.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Adds \
  --type simple \
  --set operation-type:add

$ bin/dsconfig create-request-criteria \
  --criteria-name Mods \
  --type simple \
  --set operation-type:modify

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Adds Received Any Locally" \
  --set evaluation-order-index:1 \
  --set local-level:received-any-server \
  --set "timeout:500ms" \
  --set request-criteria:Adds

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Mods Processed All Locally" \
  --set evaluation-order-index:2 \
  --set local-level:processed-all-servers \
  --set "timeout:500ms" \
  --set request-criteria:Mods
```

5. Leave server 3 with the default Replication Assurance Policy configured with no assurance levels or criteria. In practice it is common for all servers to have the same assurance levels or criteria.
6. On server 1, list the policies on the server using the `dsconfig` command to confirm that they exist on the server.

```
$ bin/dsconfig list-replication-assurance-policies
```



```
Replication Assurance Policy         : Type    : enabled : evaluation-order-index : local-level           : remote-level
-------------------------------------:---------:---------:------------------------:-----------------------:-------------
Adds Processed All Locally           : generic : true    : 1                      : processed-all-servers : none
Mods Received Any Locally            : generic : true    : 2                      : received-any-server   : none
Default Replication Assurance Policy : generic : true    : 9999                   : none                  : none
```

7. Repeat the previous step on server 2 and server 3. Server 3 should only show the Default Replication Assurance Policy.
8. Check the Replication Assurance counters on all servers before any add or modify operation using `ldapsearch`. They should be set to zero. These counters are on the replica server, which is where the policy is matched and assigned. On server 1, run the following command:

```
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-local-completed-normally: 0
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

9. Check the Replication Summary table on all of the servers. For example, on server 1, run the following command:

```
$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 0
replication-assurance-local-completed-normally: 0
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

10. Add an entry to the server 1 Directory Server. Check that the counters matched the newly added entry to the "Adds Processed All Locally" Policy and that it completed assured.

```
$ bin/ldapmodify --filename add-user.ldif --defaultAdd
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 1
replication-assurance-local-completed-normally: 1
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
replication-assurance-policy-matches: Adds Processed All Locally: 1
replication-assurance-policy-matches: Default Replication Assurance Policy:
 0
replication-assurance-policy-matches: Mods Received Any Locally: 0
replication-assurance-local-level-uses: processed-all-servers: 1
replication-assurance-remote-level-uses: none: 1

$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 1
replication-assurance-local-completed-normally: 1
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

11. Perform a modify of an entry under dc=example,dc=com on server 1. Check that the counters matched the modify operation to the "Mods Processed All Locally" policy and that the operations completed assured.

```
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 2
replication-assurance-local-completed-normally: 2
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
replication-assurance-policy-matches: Adds Processed All Locally: 1
replication-assurance-policy-matches: Default Replication Assurance Policy:
 0
replication-assurance-policy-matches: Mods Received Any Locally: 1
replication-assurance-local-level-uses: processed-all-servers: 1
replication-assurance-local-level-uses: received-any-server: 1
replication-assurance-remote-level-uses: none: 2

$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 2
replication-assurance-local-completed-normally: 2
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

You have successfully configured Assured Replication.

About the Assured Replication Controls

The LDAP SDK for Java provides an implementation of an LDAP control that can be included in add, bind, modify, modify DN, and certain extended requests to indicate the level of replication assurance desired for the associated operation. The OID for this control is 1.3.6.1.4.1.30221.2.5.28, and may have a criticality of either TRUE or FALSE.

For specific details, see the LDAP SDK javadoc for the `AssuredReplicationRequestControl` class.

Managing the Topology

The following sections describe common topology management operations.

Note: When enabling or disabling replication within a topology that contains multiple product versions, the `dsreplication` tool must be run from the server root location of a member of the topology that has the oldest product version.

To Add a Server to the Topology

The following steps assume an existing directory server topology. The commands are identical for initial enable between two servers, where one server contains data for the replication domain stored in the `userRoot` backend. If database encryption is being used on the servers in the topology, it is important that the server being initialized has a copy of the `encryption-settings` backend from the source server.

1. A majority of servers (more than 50%) in the topology and the new server, should be online.
2. Enable replication for the base DN, or base DNs, using an existing server as `host1` and the new server as `host2`.

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 austin03.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt
```

3. Optionally, compare the configurations between the two hosts used in the `dsreplication enable` command. Make sure settings are consistent across the topology and are also consistent with the new system:

```
$ bin/config-diff --sourceLocal --targetHost austin03.example.com
--targetBindDN "cn=directory manager" --targetBindPassword pass --targetPort
 1389
```

Disabling Replication and Removing a Server from the Topology

When removing a server from the topology, the remaining servers need to be made aware of the change. If the server to be removed is online, then one invocation of `dsreplication disable` is all that is necessary. If the server to be removed is offline, then two steps are required: `remove-defunct-server` from another server in the topology, and `remove-defunct-server` on the offline server to be removed. Similar to the `enable` command, more that 50% of servers not being removed from the topology need to be online during the process.

If there are additional servers that are offline and can not be online while the offline server is being removed, then it's important to make a distinction between offline servers that are permanently offline, and those that are temporarily offline. If servers are permanently offline, they should also be removed with `remove-defunct-server`. If servers are temporarily offline, once they are online, they will automatically update. The following examples show the steps in more detail:

- **Removing a server that is still online**. The `dsreplication disable` command can be run from any server, but a majority of servers in the topology need to be online.

```
$ bin/dsreplication disable --hostname austin03.example.com --port 1389 \
              --baseDN dc=example,dc=com --adminUID admin --adminPassword
 password \
              --no-prompt
```

- **Removing a server that is offline**.The `remove-defunct-server` tool can be run against any server not being removed from the topology. More than 50% of servers in the topology should be online. The `remove-defunct-server` tool can be issued after setting the JVM property `"com.unboundid.connectionutils.LdapResponseTimeoutMillis"` to change the default ten minute time out for each server to be taken out of rotation. If there are multiple servers to be removed, this can speed up the process.

```
$ bin/remove-defunct-server --serverInstanceName austin01 \
  --bindDN "cn=Directory Manager" --bindPassword password
```

  Run the `remove-defunct-server` tool on each server removed from the topology to remove any topology references.

```
$ bin/remove-defunct-server --no-prompt
```

Replacing the Data for a Replicating Domain

In the rare event that the data for the entire replication domain (such as the backend) needs to be replaced, perform the following steps:

To Replace the Data

1. With all servers online, the `dsreplication pre-external-initialization` command must be run once against any server in the topology. This stops replication for the domain. No writes are made by clients to any of the servers.

```
$ bin/dsreplication pre-external-initialization --hostname
 austin01.example.com \
   --port 1389 --baseDN dc=example,dc=com --adminUID admin \
   --adminPassword password --no-prompt
```

2. Using `import-ldif`, replace the data for `dc=example,dc=com`. Make sure that the input LDIF is free of any replication attributes by using the `--excludeReplication` option. The `--overwriteExistingEntries` option is necessary to overwrite the existing data for the domain. For example, to perform the `import-ldif` with the server offline:

```
$ bin/import-ldif --ldifFile new-data.ldif --backendID userRoot --
excludeReplication --overwriteExistingEntries
```

3. Initialize the other servers in the topology with `dsreplication initialize`, using the server which has the new data as the source host. For example:

```
$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
   --hostDestination budapest01.example.com --portDestination 1389 \
   --adminUID admin --adminPassword password --baseDN dc=example,dc=com \
   --no-prompt
```

4. Run `dsreplication post-external-initialization` once from any server in the topology. All servers in the topology must be online:

```
$ bin/dsreplication post-external-initialization --hostname
 austin01.example.com \
   --port 1389 --baseDN dc=example,dc=com --adminUID admin \
   --adminPassword password --no-prompt
```

Advanced Configuration

The following sections are advanced configuration procedures that may be appropriate for your company's deployment.

Changing the replicationChanges DB Location

You can change the `replicationChanges` DB location if on-disk space issues arise. The replication changelog database can live outside `<server-root>` and be placed in another location on the filesystem. In that case, you must specify the absolute path of the replication changelog directory.

To Change the replicationChanges DB Location

1. Use `dsconfig` to change the database location for the replication changelog, which by default is at `<server-root>/changelogDb`. The following command sets the replication changelog backend to `<server-root>/data/directory/changelogDB` . Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set "replication-db-directory:/data/directory/changelogDb" \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

2. Stop the server, move the DB files, and then restart the server.

```
$ bin/stop-server
$ mv changelogDb /data/directory
$ bin/start-server
```

Modifying the Replication Purge Delay

The replication purge delay specifies the period after which the directory server purges changes on the replication server database. Any change that is older than the purge delay is removed from the replication server database regardless of whether the change has been applied.

Currently, the PingDirectory Server sets the default purge delay to one day. Administrators can change the default purge delay using the `dsconfig` tool. To ensure proper replication processing, you must have the same purge delay value set for all replication servers in the topology.

To Modify the Replication Purge Delay

- Use `dsconfig` to change the purge delay. The property accepts time units of seconds (s), minutes (m), hours (h), days (d), or weeks (w). The following command is entered on the command line and changes the purge delay from the default one day to two days.

  In `dsconfig` interactive mode, open the **Advanced objects** menu. Select **Replication Server**. Select the replication synchronization provider, and then select the option to change the replication purge delay.

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set "replication-purge-delay:2 d"
```

Configuring a Single Listener-Address for the Replication Server

By default, the replication server binds the listening ports to all available interfaces of the machine. To bind the listener to a specific address, the address must be the hostname provided when replication is enabled and the `listen-on-all-addresses` property must be set to `FALSE`.

The replication server's configuration entry already stores a host name for itself so that it can resolve the address and specify it during the socket bind. If the server information is missing from the system, an error message will be generated with instructions on specific address binding. You can use the `dsconfig` tool to change the value of the `listen-on-all-addresses` property from TRUE (default) to FALSE.

To Configure a Replication Server to Listen on a Single Address

1. Create a new directory serverinstance with replication enabled on port 8989.
2. Run `netstat` to see the ports bound for listening on port 8989. Notice that *.8989 means that it is listening on all addresses.

```
$ netstat -an | grep LISTEN | grep 8989
```

3. Run `dsconfig` to disable listening on all addresses for the replication server.

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set listen-on-all-addresses:false
```

4. Run `netstat` again to see the ports bound for listening on port 8989. Notice that `<address>.8989` (for example, 10.8.1.211.8989) means that it is listening on the one address.

Monitoring Replication

Replication in the PingDirectory Server can be monitored the following ways:

- The `dsreplication` status subcommand displays basic information about the replicated base DNs, the number of entries replicated as well as the approximate size of replication backlogs at each Directory Server.
- The more detailed information about the state of replication can be obtained via the information exposed in its monitoring framework under `cn=monitor`. Administrators can monitor their replication topologies using several tools and protocols: the PingDataMetrics Server, SNMP, LDAP, JMX, or through the Administrative Console. See Managing Logging and Monitoring.
- The Periodic Stats Logger plug-in allows collecting replication statistics for profiling server performance. For more information, see Profiling Server Performance Using the Periodic Stats Logger.

Monitoring Replication Using cn=monitor

The `cn=monitor` branch has a number of entries that store the replication state of a topology.

- **Direct LDAP Server <baseDN> <host name:port> <serverID>**. Defines an LDAP server that is directly connected to the replication server that you are querying. The information in this entry applies to the replication server local to the `cn=monitor` entry. For detailed information, see *Summary of the Direct LDAP Monitor Information*.
- **Indirect LDAP Server <baseDN> <serverID>**. Defines an LDAP server that is connected to another replication server in the topology. While this server is connected to the same topology, it is not connected to the replication server being queried. For detailed information, see *Summary of the Indirect LDAP Server Monitor Information*.

- **Remote Repl Server <baseDN> <host name:port> <serverID>**. Defines a remote replication server that is connected to the local replication server. Information in this entry is in respect to the Replication Server local to the cn=monitor branch. For detailed information, see *Summary of the Remote Replication Server Monitor Information*.
- **Replica <baseDN>**. Stores information for an instance of the replicated naming context— also known as the replica—with respect to the Directory Server and its communication with a replication server. The Replica information is what is responsible for sending and receiving changes from the replication servers. For detailed information, see *Summary of the Replica Monitor Information*.
- **Replication Server <replPort> <serverID>**. Shows the information specific to the Replication Server running, for example, on the replication port <replPort> with a server ID of <serverID>. This entry defines the replication server. For detailed information, see *Summary of the Replication Server Monitor Information*.
- **Replication Server Database <baseDN> <serverID>**. Shows information for the changelog table of replica (suffix) in the replication server. As the Replication Server receives updates from the Directory Server, it records those changes in the changelog table. For detailed information, see *Summary of the Replication Server Database Monitor Information*.
- **Replication Server Database Environment <baseDN>**. Shows the information for the database environment for the replication server backend plus the total number of records added to and deleted from the database. For detailed information, see *Summary of the Replication Server Database Environment Monitor Information*.
- **Replication Summary <baseDN>**. Shows summary information on the replication topology and the state for a particular base DN. For detailed information, see *Summary of the Replication Summary Monitor Information*.
- **Replication Changes Backend**. Shows the backend information for all replication changes. For detailed information, see *Summary of the Replication Changes Backend Monitor Information*.
- **Replication Protocol Buffer**. Shows the state of the buffer (initially 4k) for protocol operations, which is stored in thread local storage. For detailed information, see *Summary of the Replication Protocol Buffer Monitor Information*.

Replication Best Practices

The following are recommended best practices related to replication based on our observations in actual production environments.

About the dsreplication Command-Line Utility

The following points involve some security practices as applies to replication. For specific questions, please contact your authorized support provider.

- **Developing Scripts**. The dsreplication utility maintains the history of executed dsreplication commands with the full command-line arguments in the logs/tools/dsreplication.history file. The recorded commands may be used to develop scripts to set up and configure replication.

  Scripts invoking the dsreplication utility in non-interactive mode should check the return code from the dsreplication process. A non-zero return code indicates some sort of failure.

  If output messages from the dsreplication utility are not desired, use the --quiet option to suppress them.

  The utility, by default, fails if one or more warnings are issued during the command execution. Warnings can be suppressed using the --ignoreWarnings option. For example, this option is required when using dsreplication with non-fully-qualified hostnames (for example, localhost), otherwise dsreplication will fail. In production environments, use of this flag is strongly discouraged.

  The dsreplication utility also provides an --ignoreLock option that specifies that the tool should continue processing in non-interactive mode or in scripts even if the replication topology has been locked by another invocation of dsreplication. However, this option should be used with caution.
- **Concurrent Use**. With the exception of the dsreplication status subcommand, the dsreplication subcommands cannot be executed concurrently. The command-line utility locks the replication topology at one or more servers to prevent accidental configuration changes caused by multiple dsreplication subcommands running at the same time. It is best to avoid concurrent configuration changes in general.

- **Status**. The dsreplication status subcommand requires the Replication Servers to provide monitoring information. This can lead to a delay before the output of dsreplication status is displayed. By default, dsreplication will display the status for all replicated domains (with the exception of the special domains of the schema and the server registry).

  It is recommended to select a particular base DN for dsreplication status if multiple base DNs are configured for replication.

  It is also recommended to avoid invoking dsreplication status too often (more than once every 15 seconds) or from multiple locations at the same time. Some of the information displayed by dsreplication status is based on monitor information that is not refreshed every time the monitor is queried.

  The status subcommand should not be used for collecting performance metrics. It is best to rely on replication-related information captured by the Periodic Stats Logger Plug-in.
- **Topology Tasks**. The dsreplication tool allows specifying more than one server in a topology to act as the host for other servers for the enable and initialize actions. The --topologyFilePath option can be used to specify a file with a series of hosts and ports in the topology. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available. This file is generated with the manage-topology export command.

Replication Conflicts

This section provides more in-depth information on replication conflicts than presented in earlier sections, so that administrators can understand the mechanisms and possible scenarios behind these conflicts.

Updates to Directory Server entries in a replication topology may happen independently, since replication guarantees only eventual consistency, not strong consistency. The eventual consistency model also means that conflicting changes can be applied at different directory server instances. In most cases, the Directory Server is able to resolve these conflicts automatically and in a consistent manner (i.e., all directory server instances in a replication topology will resolve each and every conflict the same way). However, in some scenarios, as seen below, manual administrative action is required. For any of these unresolved conflicts, the administrator is notified via administrative alerts.

On a high-level, the conflict resolution algorithm tries to resolve conflicts as if the operations causing the conflict in a distributed environment has been applied to a single directory server instance. For example, if the same entry is added to two different directory server instances at about the same time, then once these operations have been replicated, both directory servers will keep only the entry that was added first. The following figure highlights the differences between standalone versus replicated environments.

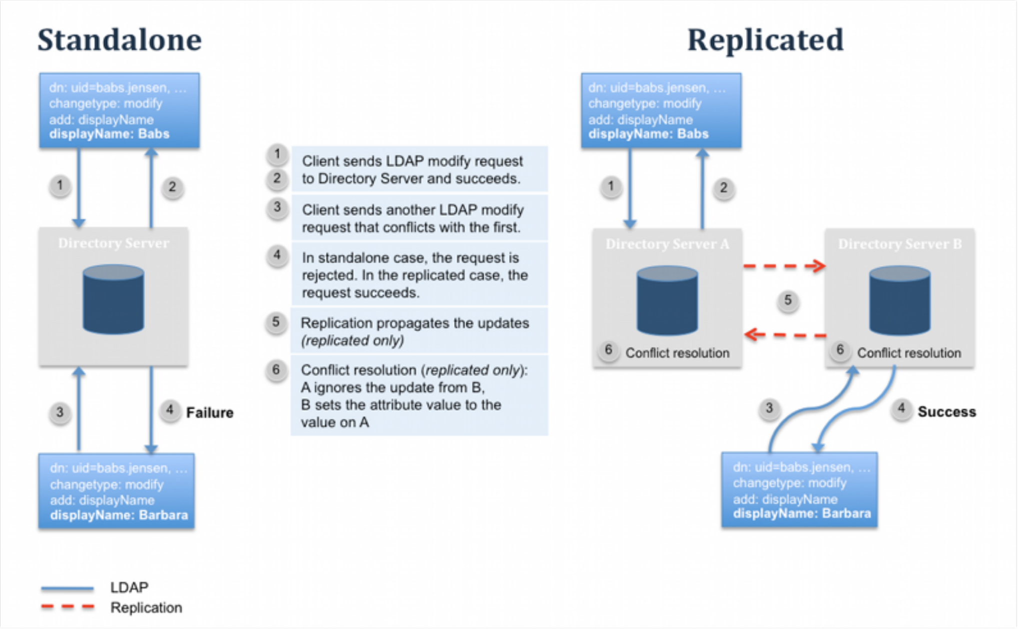

Standalone
Replicated
dn: uid=babs.jensen, …
changetype: modify
add: displayName
displayName: Babs
Directory Server
Directory Server A
Directory Server B
1
2
3
4 Failure
4 Success
5
6 Conflict resolution
dn: uid=babs.jensen, …
changetype: modify
add: displayName
displayName: Barbara
1 2 Client sends LDAP modify request to Directory Server and succeeds.
3 Client sends another LDAP modify request that conflicts with the first.
4 In standalone case, the request is rejected. In the replicated case, the request succeeds.
5 Replication propagates the updates (replicated only)
6 Conflict resolution (replicated only): A ignores the update from B, B sets the attribute value to the value on A
LDAP
Replication

Figure 12: Conflicting Operations in Standalone versus Replicated Environments

Types of Replication Conflicts

There are fundamentally two types of replication conflicts: naming and modification conflicts. Naming conflicts include operations that cause conflicts with the naming (DN) of the existing or new entries, while modification conflicts include operations that result in conflicts in the modification of attributes.

Naming Conflict Scenarios

For all of the naming conflict scenarios in the table below, assume the folloing:

- Update 1 was applied at Directory Server 1
- Update 2 was applied at Directory Server 2
- Update 1 occurred shortly before Update 2, so that Directory Server 2 received Update 1 after Update 2 was applied

The naming conflict scenario is illustrated in the following figure:

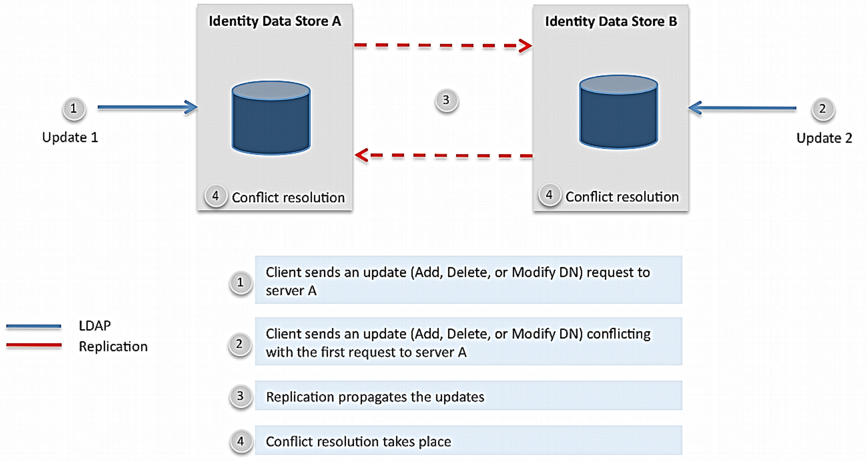

Identity Data Store A
Identity Data Store B
1 Update 1
2 Update 2
3
4 Conflict resolution
4 Conflict resolution
LDAP
Replication
1 Client sends an update (Add, Delete, or Modify DN) request to server A
2 Client sends an update (Add, Delete, or Modify DN) conflicting with the first request to server A
3 Replication propagates the updates
4 Conflict resolution takes place

Figure 13: Naming Conflict Scenario

The following table shows the result of a modification conflict depending on the type of updates that occurs. The code does not compare change sequence numbers (CSNs) but applies operations in the order they were received. This may lead to inconsistent replays.

Table 46: Naming Conflict Scenarios

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at Directory Server 2 When Update 1 is received
Modify	Delete	Yes	Modify is discarded.
Modify	Modify DN	Yes	New entry is located based on the entryUUID and Modify is applied to the renamed entry.
Delete	Delete	Yes	Delete operation is ignored.
Delete	Modify DN	Yes	Delete operation is applied to the renamed entry.
Delete of A	Add of B under A	Yes	Entry B is renamed and Entry A is deleted.
Modify DN	Delete of the same entry targeted by the Modify DN	Yes	Modify DN operation is ignored.
Modify DN with a new parent	Delete of parent	No	The entry targeted in the Modify DN operation is marked as a conflict entry.
Modify DN with a new parent	Modify DN of the parent	Yes	The entry will be moved under the new DN of the parent.
Modify DN of A with new DN B	Modify DN of C with new DN B	No	A and B will be conflict entries.
Add A	Modify DN of the parent of A	Yes	The entry is added under the new DN of the parent.
Add A	Delete of the parent of A	No	The added entry is marked as a conflict entry.
Add A	Add A with same set of attributes	Yes	The entryUUID of the incoming Add operation applied to the existing entry.
Add A	Add A with different set of attributes (or values)	No	The existing entry is marked as a conflicting entry and the incoming Add is executed.

Modification Conflict Scenarios

Modification conflicts are always resolved automatically and no manual action is required. The LDAP Modify operation allows the following modification types:

Add of one or more values
Delete of one or more values or the entire attribute
Replacement of all values

Replication does not currently support the increment LDAP modification type.

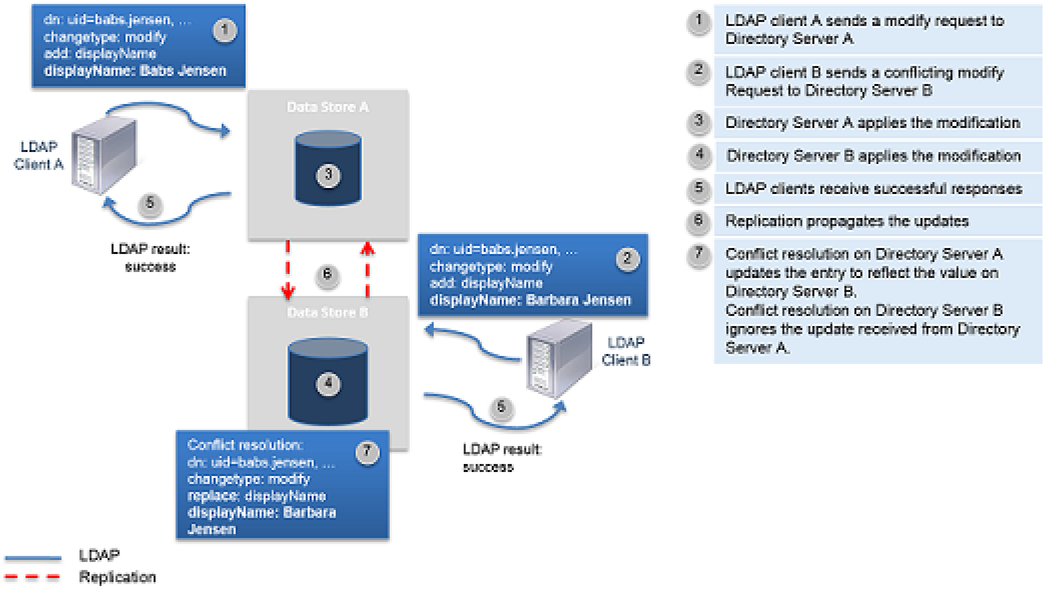
For all of the operations in the table below, assume the following:

LDAP Modify 1 was applied at Directory Server 1

LDAP Modify 2 was applied at Directory Server 2

LDAP Modify 1 occurred shortly before LDAP Modify 2, so that Directory Server 2 received LDAP Modify 1 after LDAP Modify 2 was applied.

The modification conflict scenario is illustrated in the figure below:



```
dn: uid=babs.jensen, ...
changetype: modify
add: displayName
displayName: Babs Jensen
```

```
dn: uid=babs.jensen, ...
changetype: modify
add: displayName
displayName: Barbara Jensen
```

```
Conflict resolution:
dn: uid=babs.jensen, ...
changetype: modify
replace: displayName
displayName: Barbara
Jensen
```

1. LDAP client A sends a modify request to Directory Server A
2. LDAP client B sends a conflicting modify Request to Directory Server B
3. Directory Server A applies the modification
4. Directory Server B applies the modification
5. LDAP clients receive successful responses
6. Replication propagates the updates
7. Conflict resolution on Directory Server A updates the entry to reflect the value on Directory Server B. Conflict resolution on Directory Server B ignores the update received from Directory Server A.

Figure 14: Modification Conflict Scenario

The following table shows the result of a modification conflict depending on the type of updates that occurs:

Table 47: Modification Conflict Scenarios

Modify 1	Modify 2	Result of Conflict Resolution at Directory Server 2 When Modify 1 is received
Add of a single-value attribute	Add of the same attribute with a different value	Incoming Modify is ignored.
Delete of a single-valued attribute	Replacement of the value of the same attribute	Incoming Delete is ignored.
Replacement of a single-valued attribute	Delete of the same attribute	Incoming Replacement is ignored.
Delete some values from a multi-valued attribute	Delete some values from a multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Delete of the same multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Add the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Delete value X from the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Add value Y to the same multi-valued attribute	Delete of value X is applied.
Delete value X from a multi-valued attribute	Delete value Y from the same multi-valued attribute	Delete of value X is applied.

Modify 1	Modify 2	Result of Conflict Resolution at Directory Server 2 When Modify 1 is received
Delete value X from a multi-valued attribute	Replace all values of the same multi-valued attribute	Incoming Delete is ignored.
Add of values X and Y to a multi-valued attribute	Delete value X from the same multi-valued attribute	Only value Y is added.
Delete value X from a multi-valued attribute	Add of values X and Y to the same multi-valued attribute	Incoming Delete is ignored.

Troubleshooting Replication

The following sections provide information to troubleshoot your replication deployment.

Recovering a Replica with Missed Changes

If a server has been offline for a period of time longer than the replication purge delay, the `dsreplication initialize` command must be performed to bring the replica into sync with the topology.

Server startup is the only time missed changes are detected. A missed change is a change that the replica detects that it needs, but which cannot be found within any other replication server's replicationChanges backend (stored in the path server root `/changelogDb`). If missed changes are detected, the server enters lockdown mode, where only privileged clients can make requests. Any other server that is not missing changes can be used as a source for `dsreplication initialize`.

If a manual backup and restore of the server is required, then the following steps are equivalent to `dsreplication initialize`.

Performing a Manual Initialization

In the event that an online initialization is not possible, the following steps can be used to initialize a server.

1. From another server in the replication topology, backup the `userRoot`, `adminRoot`, `schema`, `replicationChanges` backends to the `/bak` directory. If encrypted attributes are present, then the `encryption-settings` backend should also be exported. One or more encryption settings IDs may need to be exported and imported.

```
$ <source-server-root>/bin/backup --backendID userRoot -d bak/userRoot
$ <source-server-root>/bin/backup --backendID adminRoot -d bak/adminRoot
$ <source-server-root>/bin/backup --backendID schema -d bak/schema
$ <source-server-root>/bin/backup --backendID replicationChanges -d bak/
replicationChanges
$ <source-server-root>/bin/encryption-settings export --id ID --output-file
 bak/exported-key
```

2. Copy the `bak` directory to the new replica.

```
$ scp -r <source-server-root>/bak <user>@<destination-server>:<destination-
server-root>/bak
```

3. Stop the server and restore the `userRoot`, `changelog`, `adminRoot`, `replicationChanges` backends. If the `encryption-settings` backend was exported, it should also be reimported.

```
$ <destination-server-root>/bin/restore -d bak/userRoot
$ <destination-server-root>/bin/restore -d bak/adminRoot
$ <destination-server-root>/bin/restore -d bak/schema
$ <destination-server-root>/bin/restore -d bak/replicationChanges
$ <destination-server-root>bin/encryption-settings import --input-file bak/
exported-key --set-preferred
     Enter the PIN used to encrypt the definition:
```

4. Start the server using `bin/start-server`.

Fixing Replication Conflicts

Replication conflicts can occur when an incompatible change to an entry is made on two replicas at the same time. The change is processed on one replica and then replicated to the other replica, which causes the conflict. While most conflicts are resolved automatically, some require manual action.

To fix replication conflicts, initialize the replica containing the conflicts with the data from another replica that does not have conflicts. If the database is large and the number of conflicts small, running `ldapmodify` against the server with the conflict will work if the command includes the Replication Repair Control specified by OID value 1.3.6.1.4.1.30221.1.5.2. The Replication Repair Control prevents the change from replicating. It also enables changing operational attribute values, which are not normally writable.

The following steps provide an example of using the Replication Repair Control to fix replication conflicts by applying change to only the server with the conflict. There are two examples: one for a modification conflict found by performing an `ldap-diff`, and the other for a naming conflict.

To Fix a Modify Conflict

1. The `bin/ldap-diff` tool can be used to isolate conflicting entries between two replicas. The following uses the tool to search across the entire base DN for any difference in user attributes, and reports the difference in `difference.ldif`. Replace the `sourceHost` value with the server that needs the adjustment.

```
$ bin/ldap-diff --sourceHost austin02.exmple.com --sourcePort 1389 \
                        --sourceBindDN "cn=Directory Manager" --
sourceBindPassword pass \
                        --targetHost austin01.example.com --targetPort 1389
 \
                        --targetBindDN "cn=Directory Manager" --
targetBindPassword
                        --baseDN "dc=example,dc=com" --outputLDIF
 difference.ldif \
                        --searchFilter "(objectclass=*)" --numPasses 3 "*"
 pass \
                        "^userPassword"
```

2. The `difference.ldif` file is in a format that can be used with `ldapmodify` to apply changes to the server that contains conflicts. The `ldap-diff` command must have been run with the `sourceHost` value as the server with conflicts. The following is an example of the contents of `difference.ldif`:

```
dn: uid=user.1,ou=people,dc=example,dc=com
                        changetype: modify
                        add: mobile
                        mobile: +1 568 232 6789
                        -
                        delete: mobile
                        mobile: +1 568 591 7372
                        -
```

3. Run `bin/ldapmodify` to correct the entries on only the server with conflicts.

```
$ bin/ldapmodify --bindPassword password -J "1.3.6.1.4.1.30221.1.5.2" \
                        --filename difference.ldif
```

To Fix a Naming Conflict

1. In this example, a naming conflict was encountered when the replica attempted to replay an ADD of `uid=user.200,ou=people,dc=example,dc=com`. In other words, two clients added the entry at the same time as an entry of the same name was added on another replica.

   ```
   [18/Feb/2010:14:53:12 -0600] category=EXTENSIONS severity=SEVERE_ERROR
   msgID=1880359005 msg="Administrative alert type=replication-unresolved-
   conflict
   id=bbd2cbaf-90a4-42af-94a8-c1a42df32fc6
   class=com.unboundid.directory.server.replication.plugin.ReplicationDomain
   msg='An unresolved conflict was detected for DN
    uid=user.200,ou=People,dc=example,dc=com.
   The conflicting entry has been renamed to
   entryuuid=69807e3d-ab27-43a3-8759-
   ec0d8d6b3107+uid=user.200,ou=People,dc=example,dc=com'"
   ```

2. The Directory Server prepends the entryUUID to the DN of the conflicting attribute and adds a `ds-sync-conflict-entry` auxiliary object class to the entry to aid in search. For example, the following command searches for any entry that has the `ds-sync-conflict-entry` objectclass and returns only the DNs that match the filter. You should see the conflicting entry for uid=user.200.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub \
     "(objectclass=ds-sync-conflict-entry)" "1.1"
   ```

   ```
   dn: entryuuid=69807e3d-ab27-43a3-8759-
   ec0d8d6b3107+uid=user.200,ou=People,dc=example,dc=com

   dn: entryuuid=523c430e-
   a870-4ebe-90f8-9cd811946420+uid=user.200,ou=People,dc=example,dc=com
   ```

   Note: Conflict entries are not returned unless the `objectclass=ds-sync-conflict-entry` is present in the search filter.

3. After comparing the conflict entry with the target entry, the difference can be applied in a manner similar to the previous example using `ldapmodify` with the Replication Repair Control. The conflict entry can also be deleted using this command. Run `bin/ldapmodify` with the Replication Repair Control to make the fix. When making changes using the Replication Repair Control, the updates will not be propagated via replication. You should examine each and every replica one by one, and apply the necessary modifications using the request control.

   ```
   $ bin/ldapmodify -J "1.3.6.1.4.1.30221.1.5.2" \
     --filename difference.ldif
   ```

Fixing Mismatched Generation IDs

A warning that multiple generation IDs were detected for a specific suffix indicates that one or more replicas need to be re-initialized. If the warning is presented from a server after an initialization, it could be that `post-external-initialization` was not run as part of a global change in data. Running this command will fix the situation. The `dsreplication` status command will warn when any generation IDs are different across the topology.

Replication Reference

The following section shows general reference information related to replication.

Summary of the dsreplication Subcommands

A summary of the `dsreplication` subcommands and functions is presented in the table below.

Table 48: dsreplication subcommands

Subcommand	Description

cleanup-local-server	Removes replication-related artifacts from the configuration, schema as well as the server registry while the local server is offline. The subcommand does not remove references to this server from other replicas and replication servers in the topology. Therefore, it is recommended to remove this server first from the replication topology either by using the disable or remove-defunct-server subcommands. Since this subcommand can only be executed when the server is offline, replication attributes from suffixes other than the server registry or the schema will not be removed. The tool will produce an LDIF file, `logs/cleanup-backends.ldif` that may be used to the remove replica state from the base entry of these suffixes after the server is restarted.
	To remove the replication history from regular suffixes, export the formerly replicated suffixes using the --excludeReplication option of the `export-ldif` command. The resulting LDIF file can be re-imported using the `import-ldif` command. For example:
	`$ bin/export-ldif --backendID userRoot --`
	`excludeReplication \`
	`  --ldifFile cleansed.ldif`
	`$ bin/import-ldif --backendID userRoot --ldifFile`
	` cleansed.ldif`
	Exporting using the --excludeReplication option of the `export-ldif` command will also remove the replica state from the output. The LDIF created by the cleanup-local-server subcommand does not need to be applied after the server is restarted.

disable	Disables replication on the specified server for the provided base DN and removes references to this server in the other servers with which it is replicating data.
enable	Updates the configuration of the servers to replicate the data under the specified base DN. If one of the servers is already replicating the data under the base DN with other servers, executing this subcommand will update the configuration of all the servers (so it is sufficient to execute the command line once for each server you add to the replication topology).
initialize	Initializes the data under the specified base DN on the destination server with the contents on the source server.
initialize-all	Initializes the data under the specified base DN on all servers in the replication topology with the contents on the specified server.
post-external-initialization	Used with `pre-external-initialization`, the command resets the generation ID based on the newly-loaded data. This subcommand must be called after initializing the contents of all the replicated servers using the `import-ldif` tool or `dsreplication initialize`. Specify the list of base DNs that have been initialized and provide the credentials of any of the servers that are being replicated. See the usage of the pre-external-initialization subcommand for more information. This subcommand only needs to be run on one of the replicas once.
pre-external-initialization	Clears the existing generation ID and removes all accumulated changes of the replicated suffix from the replication changelog database at each and every replication server. This subcommand should be used when globally restoring the replicas on all of the servers in a topology. You must specify the list of base DNs that will be initialized and provide the credentials of any of the servers that are being replicated. After calling this subcommand, initialize the contents of all the servers in the topology, then call the

Subcommand	Description
	post-external-initialization subcommand. This subcommand only needs to be run on one of the replicas once.
remove-defunct-server	Removes an offline defunct server from the replication on all servers in the topology.
status	Displays the status of replication domains. If no base DNs are specified as parameters, the information for all base DNs is displayed. Available options with the status subcommand are: --showAll, --displayServerTable, --location.

Summary of the Direct LDAP Monitor Information

The following table provides a description of the attributes in the `cn=Direct LDAP Server` monitor entry. The DN for the monitor entry is as follows:

```
dn: cn=Direct LDAP Server <baseDN> <host name:ldapPort> <serverID>,cn=monitor
```

Table 49: Direct LDAP Monitor Information

Monitor Attribute	Description
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this server is connected. The first number is the replication server port number and the second number is the server-id of the replication server.
replica-id:<serverID>	Replica ID number.
replication-backlog	Number of changes that the replication server has not seen from the server.
missing-changes	Number of missing changes.
approximate-delay	Difference between the time of the last change that the replication server has seen from the LDAP server and the most current time stamp on the latest change on the server.
base-dn	Base DN
ssl-encryption	Flag to indicate if SSL encryption is in use.
protocol-version	Displays the replication protocol version.
generation-id	Generation ID for the base DN on the Directory Server.
restricted	Boolean that indicates whether the replication domain is restricted in an Entry Balancing Configuration with the Directory Proxy Server.
ack-sent	Number of acknowledgement messages sent to this replica (not currently used).
ack-received	Number of acknowledgement messages received from this replica (not currently used).
add-sent	Number of protocol messages with an LDAP Add sent to this replica.
add-received	Number of protocol messages with an LDAP Add received from this replica.
delete-sent	Number of protocol messages with an LDAP Delete sent to this replica.
delete-received	Number of protocol messages with an LDAP Delete received from this replica.
done-sent	Number of done messages sent to this replica. A done message indicates an end of online initialization session. If the value is non-zero, then this replica has been initialized over the replication protocol.

Monitor Attribute	Description
done-received	Number of done messages received from this replica. A done message indicates an end of online initialization session. If the value is non-zero, then this replica has completed the initialization of other replicas over the replication protocol.
entry-sent	Number of entry messages sent to this replica. Entry messages carry replicated data to initialize replicas over the replication protocol.
entry-received	Number of entry messages received from this replica. Entry messages carry replicated data to initialize replicas over the replication protocol.
error-sent	Number of error messages sent to this replica.
error-received	Number of error messages received from this replica.
heartbeat-sent	Number of heartbeat messages sent to this replica.
heartbeat-received	Number of heartbeat messages received from this replica (should always be 0, since the replica never sends a heartbeat message to the server).
initialize-request-sent	Number of initialize-request messages sent to this replica. This message is sent when another replica requested initialization of its data using the replication protocol.
initialize-request-received	Number of initialize-request messages received from this replica. This message is sent when this replica requested initialization of its data using the replication protocol from another replica.
initialize-target-sent	Number of initialize-target messages sent to this replica. This message is sent before another replica has started the initialization of this replica.
initialize-target-received	Number of initialize-target messages received from this replica. This message is sent before this replica starts the initialization of one or more replicas.
modify-sent	Number of protocol messages with an LDAP Modify sent to this replica.
modify-received	Number of protocol messages with an LDAP Modify received from this replica.
modify-dn-sent	Number of protocol messages with an LDAP Modify DN sent to this replica.
modify-dn-received	Number of protocol messages with an LDAP Modify DN received from this replica.
repl-server-start-sent	Number of replication-server-start messages sent to this replica (should never be more than 1). The Replication Server responds with this message to the start message received from the replica.
repl-server-start-received	Number of replication-server-start messages received from this replica (should always be 0).
reset-generation-id-sent	Number of reset generation ID messages received from this replica.
reset-generation-id-received	Number of reset generation ID messages sent to this replica (should always be 0).
server-start-sent	Number of server-start messages sent to this replica (should always be 0).
server-start-received	Number of server-start messages received from this replica (should never be more than 1). Server-start is the first message the replica sends after establishing a replication connection.

Monitor Attribute	Description
window-sent	Number of window messages sent to this replica. Window messages are used for cumulative acknowledgement in the replication protocol.
window-received	Number of window messages received from this replica. Window messages are used for cumulative acknowledgement in the replication protocol.
window-probe-sent	Number of window probe messages sent to this replica (should always be 0).
window-probe-received	Number of window probe messages received from this replica. The replica sends a window probe message to the server if the send window in the replica is closed and the replica is unable to publish updates to the server.
update-sent	Number of changes sent to this server.
update-received	Number of changes received from this server.
internal-connection	Indicates if the replica is in the same process as the replication server.
server-state	Displays the state of the replica. Displays the latest change number that the replica has seen from all the other replicas including itself.
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
max-send-window	Maximum number of changes that can be sent to the LDAP server before requiring an ACK.
current-send-window	Current number of changes remaining to be sent to the LDAP server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received by the LDAP server before sending an ACK.
current-rcv-window	Number of changes remaining to be received by the LDAP server before sending an ACK Server.
degraded	Indicates that the generation ID of the replica does not match the generation ID of the server. This is a temporary state, when loading data into the topology or the replica has not been initialized. Normally, it should be false.

Summary of the Indirect LDAP Server Monitor Information

The following table provides a description of the attributes in the `cn=Indirect LDAP Server` monitor entry. These attributes provide information about a Directory Server that is connected to a different replication server in the topology.

```
dn: cn=Indirect LDAP Server <baseDN> <serverID>,cn=monitor
```

Table 50: Indirect LDAP Server Monitor Information

Monitor Attribute	Description
replica-id: <serverID>	ID number identifying the replica.
base-dn: <baseDN>	Base DN
connected-to: Remote Repl Server <baseDN> <host name:replPort> <replID>	Replication server to which the directory server is connected.
replication-backlog	Number of changes that the replication server has not seen from the server.
approximate-delay	Amount of time between the last change seen by this Directory Server and the most recent change seen by the remote replication server. This value is the amount of time between the time stamps, not the amount of time required to synchronize the two servers.
generation-id	Generation ID for this suffix on this remote replication server.
consumed-update-recent-rate	Rate that the connected Replication Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Replication Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.

Summary of the Remote Replication Server Monitor Information

The following table provides a description of the attributes in the `cn=Remote Repl Server` monitor entry. The DN for the monitor entry is as follows:

```
dn: cn=Remote Repl Server <baseDN> <host name:replPort> <serverID>,cn=monitor
```

Table 51: Remote Replication Server Monitor Information

Monitor Attribute	Description
replication-server: <host name>:<repl port>	Host name and replication port number of the Replication Server.
replication-server-id:<serverID>	Server ID for the Replication Server.
available	Indicates if the remote replication server is available or not. Values: true or false.
sending-paused	Indicates if sending is paused. Values: true or false.
receiving-paused	Indicates if receiving is paused. Values: true or false.
wan-gateway-priority	Specifies the WAN Gateway priority of the remote replication server.
is-wan-gateway	Indicates if the remote replication server is a WAN Gateway. Values: true or false.
wan-gateway-desired	Indicates if the remote replication server is a desired gateway. Values: true or false. This entry together with the `is-wan-gateway` property indicates the desired state. **is-wan-gateway=false, wan-gateway-desire=false**: Indicates another server with a higher gateway priority exists or the gateway priority is set to disabled.

Monitor Attribute	Description
	is-wan-gateway=false, wan-gateway-desire=true: Indicates that the remote replication server wants to be a WAN gateway. **is-wan-gateway=true, wan-gateway-desire=false**: Indicates that the remote replication server wants to give up its role as a WAN gateway. **is-wan-gateway=false, wan-gateway-desire=true**: Indicates the remote replication server wants to remain as a gateway server.
base-dn	base DN
ssl-encryption	Flag to indicate if SSL encryption is in use.
protocol-version	Replication protocol version.
generation-id	Generation ID for this suffix on this remote replication server.
restricted	Indicates that remote replication server is in an entry-balancing deployment.
add-sent	Number of protocol messages with an LDAP Add sent to the remote replication server.
add-received	Number of protocol messages with an LDAP Add received from the remote replication server.
delete-sent	Number of protocol messages with an LDAP Delete sent to the remote replication server.
delete-received	Number of protocol messages with an LDAP Delete received from the remote replication server.
done-sent	Number of done messages sent to the remote replication server. A done message indicates an end of online initialization session.
done-received	Number of done messages received from the remote replication server. A done message indicates an end of online initialization session.
entry-sent	Number of entry messages sent to the remote replication server. Entry messages carry replicated data to initialize replicas over the replication protocol.
entry-received	Number of entry messages received from the remote replication server. Entry messages carry replicated data to initialize replicas over the replication protocol.
error-sent	Number of error messages sent to the remote replication server.
error-received	Number of error messages received from the remote replication server.
heartbeat-sent	Number of heartbeat messages sent to the remote replication server.
heartbeat-received	Number of heartbeat messages received from the remote replication server.
initialize-request-sent	Number of initialize-request messages sent to the remote replication replication server. This message is used during online initialization from one replica to another.
initialize-request-received	Number of initialize-request messages received from the remote replication server. This message is used during online initialization from one replica to another.
initialize-target-sent	Number of initialize-target messages sent to the remote replication server. This message is used before online initialization from one replica to another.

Monitor Attribute	Description
initialize-target-received	Number of initialize-target messages received from the remote replication server. This message is used before online initialization from one replica to another.
modify-sent	Number of protocol messages with an LDAP Modify sent to the remote replication server.
modify-received	Number of protocol messages with an LDAP Modify received from the remote replication server.
modify-dn-sent	Number of protocol messages with an LDAP Modify DN sent to the remote replication server.
modify-dn-received	Number of protocol messages with an LDAP Modify DN received from the remote replication server.
monitor-sent	Number of monitor messages sent to the remote replication server. This message is primarily used when communicating with directory servers running a prior release.
monitor-received	Number of monitor messages received from the remote replication server. This message is primarily used when communicating with directory servers running a prior release.
monitor-request-sent	Number of monitor requests sent to the remote replication server. The receiving server will respond with a monitor message that includes server's information about the state of the topology.
monitor-request-received	Number of monitor requests received from the remote replication server. This server will respond with a monitor message that includes server's information about the state of the topology.
monitor-v2-sent	Number of monitor messages sent to the remote server. This monitor message is only used with Directory Server v3.5 or later.
monitor-v2-received	Number of monitor messages received from the remote server. This monitor message is only used with Directory Server v3.5 or later.
pause-sending-updates-sent	Number of pause-sending-updates messages sent to the remote server. The remote server must stop sending update messages to this server when receiving this message.
pause-sending-updates-received	Number of pause-sending-updates messages received from the remote server. This server will stop sending update messages to the remote server upon receiving this message.
repl-server-start-sent	Number of replication-server-start messages sent to the remote replication server (should never be more than 1). The Replication Server responds with this message to the replication-server-start message received from remote replication servers.
repl-server-start-received	Number of replication-server-start messages received from the remote replication server (should never be more than 1) window-sent: the number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
reset-generation-id-sent	Number of reset generation ID messages received from the remote replication server. This message is sent before and after the data is initialized in the topology.

Monitor Attribute	Description
reset-generation-id-received	Number of reset generation ID messages sent to the remote replication server. This message is sent before and after the data is initialized in the topology.
server-info-sent	Number of replication server information messages sent to the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
server-info-received	Number of replication server information messages received from the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
set-source-location-sent	Number of set-source-locations messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
set-source-location-received	Number of set-source-locations messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
start-sending-updates-sent	Number of start-sending-updates messages sent to the remote replication server. The remote server may only start sending updates to this server after receiving this message.
start-sending-updates-received	Number of start-sending-updates messages received from the remote server. Sending update messages to the remote server may only start after receiving this message.
window-sent	Number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
window-received	Number of window messages received from the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
messages-sent	Total number of messages sent.
messages-received	Total number of messages received.
update-sent	Total number of updates sent.
update-received	Total number of updates received.
server-state	Displays the server state of the remote replication server. It displays the last change seen on the remote replication server.
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.

Monitor Attribute	Description
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
max-send-window	Maximum number of changes that can be sent to the remote replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the remote replication server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received from the remote replication server before sending an ACK.
current-rcv-window	Number of changes remaining to be received from the remote replication server before sending an ACK.
degraded	Indicates that the generation ID of the replica does not match the generation ID of the remote replication server. This is a temporary state, when loading data into the topology or the replica has not been initialized. Normally, it should be false.

Summary of the Replica Monitor Information

The following table provides a description of the attributes in the `cn=Replica` monitor entry for a specific base DN.

```
dn: cn=Replica <baseDN>,cn=monitor
```

Table 52: Indirect LDAP Server Monitor Information

Monitor Attribute	Description
base-DN: <baseDN>	Specified base DN. The monitor entries track your company's base DN (or `dc=example,dc=com`), `cn=schema`, and `cn=topology,cn=config`.
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this LDAP Server is connected. The first number is the replication server port number and the second number is the `server-id` of the replication server.
lost-connections	Number of times the Directory Server has lost connection to a replication server.
received-updates	Number of updates that the Directory Server Replica has received from the connected replication server.
sent-updates	Number of updates sent to the replication server.
pending-updates	Number of updates pending to send to the replication server.
replayed-updates	Total number of updates from the replication server that have been replayed for this replica.
replayed-updates-ok	Number of updates for this replica that have been successfully replayed with no conflicts.
replayed-update-failed	Number of updates for this replica that were successfully replayed after automatically resolving a modify conflict.
resolved-modify-conflicts	Number of updates for this replica that were successfully resolved after a modify conflict.

Monitor Attribute	Description
resolved-naming-conflicts	Number of updates for this replica that were successfully resolved after a naming conflict.
unresolved-naming-conflicts	Number of updates for this replica that could not be replayed due to an unresolvable naming conflict.
replica-id	Server ID for this replica.
max-rcv-window	Maximum number of changes that the Directory Server Replica can receive at a time before sending an acknowledgment back to the replication server.
current-rcv-window	Current received window size for this replica.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-rcv-window	Number of changes remaining to be received from the replication server before it must send an ACK.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the replication server before requiring an ACK.
ssl-encryption	Flag to indicate if SSL encryption is in use.
generation-id	Generation ID for this suffix on the Directory Server.
replication-backlog	Number of changes that are from this replica.

Summary of the Replication Server Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server` monitor entry.

```
dn: cn=Replication Server <baseDN> <replServerID>,cn=monitor
```

Table 53: Replication Server Monitor Information

Monitor Attribute	Description
replication-server-id	Server ID for the Replication server ID.
replication-server-port	Port number on which the replication server listens for communication from other servers.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
Generation IDs by Base DN	List of generation IDs for each base DN on the server.
num-outgoing-replication-server-connections	Number of outgoing connections from the replication server.
num-incoming-replication-server-connections	Number of incoming connections into the replication server.
num-incoming-replica-connections	Number of incoming connections to the replica.

Summary of the Replication Server Database Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server database` monitor entry.

```
dn: cn=Replication Server database <baseDN> <replServerID>,cn=monitor
```

Table 54: Replication Server Database Monitor Information

Monitor Attribute	Description
database-replica-id: <replicaID>	Specifies the replication server ID.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
first-change	First change number that is in this replication database table for this suffix from this `server-id`. For example, an example entry looks like the following: `0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011`
last-change	Last change number that is in this replication database table for this suffix from this `server-id`. For example, an example entry looks like the following: `0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011`
queue-size	Number of changes in the replication server queue waiting to be sent to this remote replication server database.
queue-size-bytes	Size in bytes of all the messages waiting in the queue.
records-added	Displays the number of records changed or added to the DIT.
records-removed	Displays the number of records removed from the DIT.

Summary of the Replication Server Database Environment Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server Database Environment` monitor entry, which includes the environment variables associated with the Oracle Berkeley Database Java Edition backend.

```
dn: cn=Replication Server Database Environment,cn=monitor
```

Table 55: Replication Server Database Environment Monitor Information

Monitor Attribute	Description
je-version	Current version of the Oracle Berkeley Java Edition.
current-db-cache-size	Current DB cache size.
max-db-cache-size	Maximum DB cache size.
db-cache-percent-full	Percentage of the cache used by the Directory Server.
db-directory	Directory that holds the changelogDb file.
db-on-disk-size	Size of the DB on disk.
cleaner-backlog	Number of log files that must be cleaned for the cleaner to meet its target utilization.
random-read-count	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.

Monitor Attribute	Description
random-write-count	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
sequential-read-count	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
sequential-write-count	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
nodes-evicted	Accumulated number of nodes evicted.
active-transaction-count	Number of currently active transactions.
num-checkpoints	Number of checkpoints. A checkpoint is a process that writes to your log files all the internal BTree nodes and structures modified as a part of write operations to your log files to facilitate a quick recovery.
checkpoint-in-progress: false	Indicates if a checkpoint is in progress.
total-checkpoint-duration-millis	Total time in milliseconds for all checkpoints.
average-checkpoint-duration-millis	Average time in milliseconds for all checkpoints.
last-checkpoint-duration-millis	Duration in milliseconds of the last checkpoint run.
last-checkpoint-start-time	Start time of the last checkpoint.
last-checkpoint-stop-time	Stop time of the last checkpoint.
millis-since-last-checkpoint	Time in milliseconds since the last checkpoint.
read-locks-held	Total read locks currently held.
write-locks-held	Total write locks currently held.
transactions-waiting-on-locks	Total transactions waiting for locks
je-env-stat-AdminBytes	Number of bytes of JE cache used for log cleaning metadata and other administrative structure.
je-env-stat-BufferBytes	Total memory currently consumed by log buffers, in bytes.
je-env-stat-CacheDataBytes	Total memory of cache used for data.
je-env-stat-CacheTotalBytes	Total amount of JE cache in use, in bytes.
je-env-stat-CleanerBacklog	Number of files to be cleaned to reach the target utilization.
je-env-stat-CursorsBins	Number of bottom internal nodes (BINs) encountered by the INCompressor that had cursors referring to them when the compressor ran. The compressor thread cleans up the internal BTree as records are deleted to ensure unused nodes are not present.
je-env-stat-DataBytes	Amount of JE cache used for holding data, keys and internal Btree nodes, in bytes.
je-env-stat-DbClosedBins	Number of bins encountered by the INCompressor that had their database closed between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-EndOfLog	Location of the next entry to be written to the log.
je-env-stat-InCompQueueSize	Number of entries in the INCompressor queue when the getStats() call was made.
je-env-stat-LastCheckpointEnd	Location in the log of the last checkpoint end.

Monitor Attribute	Description
je-env-stat-LastCheckpointId	ID of the last checkpoint.
je-env-stat-lastCheckpointStart	Location in the log of the last checkpoint start.
je-env-stat-lockBytes	Number of bytes of JE cache used for holding locks and transactions.
je-env-stat-NBINsStripped	Number of BINS stripped by the evictor.
je-env-stat-NCacheMiss	Total number of requests for database objects which were not in memory.
je-env-stat-NCheckpoints	Total number of checkpoints run so far.
je-env-stat-NCleanerDeletions	Number of cleaner file deletions this session.
je-env-stat-NCleanerEntriesRead	Accumulated number of log entries read by the cleaner.
je-env-stat-NCleanerRuns	Number of cleaner runs this session.
je-env-stat-NClusterLNsProcessed	Accumulated number of leaf nodes (LNs) processed because they qualify for clustering.
je-env-stat-NDeltaINFlush	Accumulated number of delta internal nodes (INs) flushed to the log.
je-env-stat-NEvictPasses	Number of passes made to the evictor.
je-env-stat-NFSyncRequests	Number of fsyncs requested through the group commit manager. `Fsync()` synchronizes the filesystem after a write or a transaction.
je-env-stat-NFSyncTimeouts	Number of fsync requests submitted to the group commit manager which timed out.
je-env-stat-NFSyncs	Number of fsyncs issued through the group commit manager.
je-env-stat-NFileOpens	Number of times a log file has been opened.
je-env-stat-NFullBINFlush	Accumulated number of full bottom internal nodes (BINS) flushed to the log.
je-env-stat-NFullINFlush	Accumulated number of full INs flushed to the log.
je-env-stat-NINsCleaned	Accumulated number of INs cleaned.
je-env-stat-NINsDead	Accumulated number of INs that were not found in the tree anymore (deleted).
je-env-stat-NINsMigrated	Accumulated number of INs migrated.
je-env-stat-NINsObsolete	Accumulated number of INs obsolete.
je-env-stat-NLNQueueHits	Accumulated number of LNs processed without a tree lookup.
je-env-stat-NLNsCleaned	Accumulated number of LNs cleaned.
je-env-stat-NLNsDead	Accumulated number of LNs that were not found in the tree anymore (deleted).
je-env-stat-NLNsLocked	Accumulated number of LNs encountered that were locked.
je-env-stat-NLNsMarked	Accumulated number of LNs that were marked for migration during cleaning.
je-env-stat-NLNsMigrated	Accumulated number of LNs migrated.
je-env-stat-NLNsObsolete	Accumulated number of LNs obsolete.
je-env-stat-NLogBuffers	Number of log buffers currently instantiated.
je-env-stat-NMarkedLNsProcessed	Accumulated number of LNs processed because they were previously marked for migration.

Monitor Attribute	Description
je-env-stat-NNodesExplicitlyEvicted	Accumulated number of nodes evicted.
je-env-stat-NNodesScanned	Accumulated number of nodes scanned to select the eviction set.
je-env-stat-NNodesSelected	Accumulated number of nodes selected to evict.
je-env-stat-NNotResident	Number of requests for database objects not contained within the in memory data structures.
je-env-stat-NOpenFiles	Number of files currently open in the file cache.
je-env-stat-NPendingLNsLocked	Sccumulated number of pending LNs that could not be locked for migration because of a long duration application lock.
je-env-stat-NPendingLNsProcessed	Accumulated number of LNs processed because they were previously locked.
je-env-stat-NRandomReadBytes	Number of bytes read which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomReads	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWriteBytes	Number of bytes written which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWrites	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NRepeatFaultReads	Number of reads which had to be repeated when faulting in an object from disk because the read chunk size controlled by `je.log.faultReadSize` is too small.
je-env-stat-NRepeatIteratorReads	Number of times we try to read a log entry larger than the read buffer size and can't grow the log buffer to accommodate the large object.
je-env-stat-NRootNodesEvicted	Accumulated number of database root nodes evicted.
je-env-stat-NSequentialReadBytes	Number of bytes read which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialReads	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWriteBytes	Number of bytes written which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWrites	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NSharedCacheEnvironments	Number of environments using the shared cache.
je-env-stat-NTempBufferWrites	Number of writes which had to be completed using the temporary marshalling buffer because the fixed size log buffers specified by `je.log.totalBufferBytes` and `je.log.numBuffers` were not large enough.
je-env-stat-NToBe-CleanedLNsProcessed	Accumulated number of LNs processed because they are soon to be cleaned.
je-env-stat-NonEmptyBins	Number of non-empty bins.
je-env-stat-ProcessedBins	Number of bins that were successfully processed by the INCompressor.

Monitor Attribute	Description
je-env-stat-RequiredEvictBytes	Number of bytes that must be evicted to get within the memory budget.
je-env-stat-SharedCacheTotalBytes	Total amount of the shared JE cache in use, in bytes.
je-env-stat-SplitBins	Number of bins encountered by the INCompressor that were split between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-TotalLogSize	Approximation of the current total log size in bytes.
je-env-stat-NOwners	Total lock owners in the lock table.
je-env-stat-NReadLocks	Total read locks currently held.
je-env-stat-NRequests	Total number of lock requests to date.
je-env-stat-NTotalLocks	Total locks currently in the lock table.
je-env-stat-NWaiters	Total transactions waiting for locks.
je-env-stat-NWaits	Total number of lock waits to date.
je-env-stat-NWriteLocks	Total write locks currently held.
je-env-stat-LastCheckpointTime	Time of the last checkpoint.
je-env-stat-LastTxnId	Last transaction ID allocated.
je-env-stat-NAborts	Number of transactions that have aborted.
je-env-stat-NActive	Number of transactions that are currently active.
je-env-stat-NBegins	Number of transactions that have begun.
je-env-stat-NCommits	Number of transactions that have committed.
je-env-stat-NXAAborts	Number of XA transactions that have aborted.
je-env-stat-NXACommits	Number of XA transactions that have committed.
je-env-stat-NXAPrepares	Number of XA transactions that have been prepared.

Summary of the Replication Summary Monitor Information

The following table provides a description of the attributes in the `cn=Replication Summary` monitor entry.

```
dn: cn=Replication Summary <baseDN>,cn=monitor
```

Table 56: Replication Summary Monitor Information

Monitor Attribute	Description
base-dn:<baseDN>	Base DN summary.
replica: replica-id, ldap-server connected-to, generation-id, replication-backlog, recent-update-rate, peak-update-rate, age-of-oldest-backlog-change	Summary information for each replica in the topology. This entry appears for each replica in the topology with its own respective properties.
replication-server: server-id, server, generation-id, status, last-connected, last-failed, failed-attempts, attributes.	Summary information for each remote replication server only in the topology. This entry appears for each Replication Server in the topology with its own respective `serverID` and `server` properties.
update-queue: id, max-count, current-count, max-size, current-	Summary information for each update queue on a server:

Monitor Attribute	Description
size, polling-source, polling-source-changed	**id**. The ID of the receiving replica or replication server **max-count**. The maximum number of update messages that the sending replication server will keep in memory for the receiving replica or replication server. If the receiver cannot accept messages fast enough for any reason (high load, network latency, etc), then this queue will fill up. When that happens, the sending replication server will read update messages from the changelog backend. This slows down the update processing considerably. **current-count**. The number of update messages currently on the queue that have not been sent to the receiving replica or replication server. Every time the sending replication server sends an update to the receiving replica or replication server, this counter is decremented. **max-size**. The maximum total size (in bytes) of update messages that may be in the queue. This queue is capped by both the maximum count (`max-count`) and the `max-size` setting, whichever is reached first. **current-size**. The total size of update messages currently on the queue that have not been sent to the receiving replica or replication server. Every time the sending replication server sends an update to the receiving replica or replication server, this value is decremented by the size of the published update message. **polling-source**. Either 'memory' or 'db'. If set to 'memory', the sending replication server relies only on the in-memory queue to push update messages to the receiving replica or replication server. If set to 'db', update messages are read and sorted from the changelog database, which is significantly slower than publishing updates from the in-memory queue. **polling-source-changed**. The total number of times the polling-source attribute has changed value (either from 'db' to 'memory' or from 'memory' to 'db'). If this value changes very frequently, then the queue size setting is probably too low.

Summary of the replicationChanges Backend Monitor Information

The following table provides a description of the attributes in the `cn=replicationChanges Backend` monitor entry.

```
dn: cn=replicationChanges Backend,cn=monitor
```

Table 57: replicationChanges Backend Monitor Information

Monitor Attribute	Description
ds-backend-id	ID descriptor for the backend. Typically, this will be "replicationChanges".
ds-backend-base-dn	Base DN for the backend. Typically, this will be `cn=replicationChanges`.
ds-backend-is-private	Flag to indicate if the backend is private.
ds-backend-entry-count	Entry count for the backend.
ds-base-dn-entry-count	Entry count for the base DN and the specified base DN.
ds-backend-writability-mode	Flag to indicate if the backend is writable or not.

Summary of the Replication Protocol Buffer Monitor Information

The following table provides a description of the attributes in the `cn=Replication Protocol` Buffer monitor entry. The monitors provide information on the state of the buffer for protocol operations, which is kept in the local storage.

```
dn: cn=Replication Protocol Buffer,cn=monitor
```

Table 58: Replication Protocol Buffer Monitor Information

Monitor Attribute	Description
saved-buffers	Number of protocol buffers. Initial buffer size is 4k.
reallocations	Number of times the buffers had to be reallocated due to insufficient size.
large-buffer-creates	Number of times a buffer larger than 512k was requested.
large-buffer-evictions	Number of times a buffer was removed from the thread local storage, because it has grown above 512k.

Advanced Topics Reference

This chapter presents background reference information covering advanced replication topics.

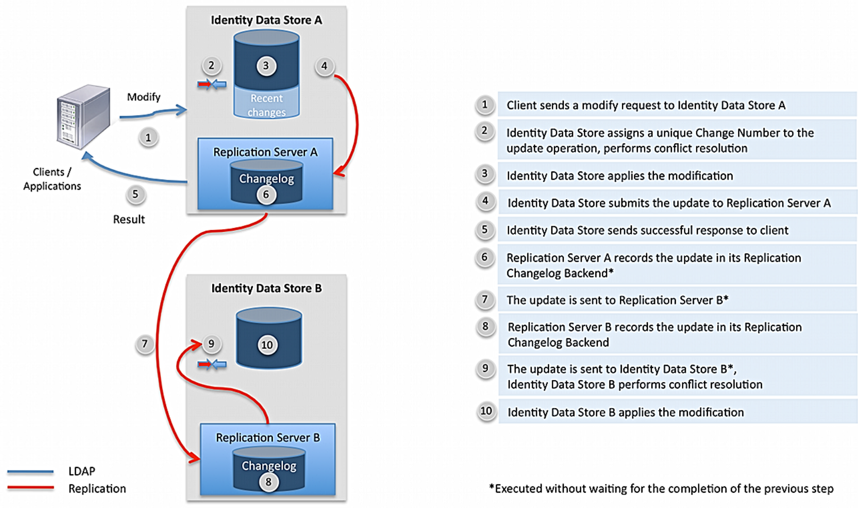
About the Replication Protocol

Replication communicates using a proprietary binary protocol that is implemented on top of the TCP/IP protocol using SSL encryption. Some protocol messages are used for administrative purposes (such as WAN Gateway server negotiation or flow control), some carry updates to replicated data, while others are directed to all servers for monitoring requests.

In a replicated topology, each participating Directory Server is connected to every other server via the replication server port in order to monitor health. Servers which share the same location setting are also connected to rapidly replicate changes and lastly the WAN Gateway servers are all interconnected to replicate changes across locations.

Directory Servers keep connections open as long as possible to reduce the communication latency when messages are exchanged. Heartbeat messages are transmitted on a regular basis to detect a network failure or an unresponsive directory server as early as possible. Heartbeat messages also prevent idle connections from being closed by firewalls.

The following detailed communication flow will be used to describe major components of replication. This illustration is the expanded view of figure shown in the Overview section.



1 Client sends a modify request to Identity Data Store A

2 Identity Data Store assigns a unique Change Number to the update operation, performs conflict resolution

3 Identity Data Store applies the modification

4 Identity Data Store submits the update to Replication Server A

5 Identity Data Store sends successful response to client

6 Replication Server A records the update in its Replication Changelog Backend*

7 The update is sent to Replication Server B*

8 Replication Server B records the update in its Replication Changelog Backend

9 The update is sent to Identity Data Store B*, Identity Data Store B performs conflict resolution

10 Identity Data Store B applies the modification

*Executed without waiting for the completion of the previous step

Figure 15: Replication Communication Flow

Step 1. Client sends a Modify request to Directory Server A.

Step 2. Directory Server A assigns a unique change number to the operation. Conflict resolution is executed to see if the Modify request is in conflict with the existing attribute types or values in the existing entry. The change number is assigned before the Directory Server backend is updated so that the arrival order of client requests can be preserved. Historical data in the target entry is updated to reflect the change. Note that historical data is only updated for ADD and MODIFY operations.

Step 3. Directory Server applies the modifications in the corresponding backend.

Step 4. If the MODIFY operation successfully completes, then the Directory Server will submit the update to its embedded *Replication Server*. The Replication Server is a component within the Directory Server process responsible for propagating updates to and from the replication topology. The Directory Server itself only communicates with a single replication server, whereas the replication server component is connected to all other replication servers. If the Directory Server process exits unexpectedly and some updates have not been passed to the Replication Server, the backend has the ability to recover the last 50,000 recent changes that were processed at this server, guaranteeing that these changes can be replicated when the server starts up. The figure above also shows that replication protocol is used not just between replication servers but also between the Directory Server and the Replication Server.

Step 5. The response is sent to the client. In this example, a successful response is assumed.

Step 6. The Replication Server records the update in its own Changelog backend (i.e., backend ID of `replicationChanges`) and on disk with the path to `changelogDb` under the server root. The Replication Changelog backend keeps track of updates at each and every Directory Server in the replication topology. When a Directory Server joins the replication topology after being disconnected for some reason, updates from the Replication Changelog backend are re-sent to this Directory Server. Old records from the Replication Changelog backend are purged, which by default removes records older than 24 hours. If the backend does not contain all of the records that another Directory Server needed when rejoining the replication topology, then the replicated data set in the Directory Server must be re-initialized. In this case, the Directory Server enters lockdown mode and an administrative alert is sent.

Step 7. The Replication Server submits the update to the replication server component in Directory Server B. If there were more Directory Servers in this example, the Replication Server would submit the update to all the other replication servers in the same location.

Step 8. Just like in Step 6, the Replication Server component receiving an update inserts the change into its Replication Changelog backend.

Step 9. The update is forwarded to the Replica in Directory Server B. Conflict resolution is executed to see if the Modify request is in conflict with the existing attribute types or values in the existing entry.

Step 10. The Directory Server applies the modification in the corresponding backend. The Recent Changes database is not updated, because only updates that originated at this Directory Server are recorded in the Recent Changes database.

Change Number

As seen in the previous Figure, the Directory Server assigns a unique change number to each update operation (specifically, ADD, DELETE, MODIFY, or MODIFY DN operations) to track each request when received from a client. The change number not only identifies each and every update, but it also allows ordering updates to the replicated data the same way on each Directory Server. The change number is composed of the following multiple fields:

- **Timestamp** that identifies when the update was made. The timestamp is time-zone-independent and recorded with millisecond resolution.
- **Server ID** that uniquely identifies the Directory Server where the update was made.
- **Sequence number** that defines the order of the updates received from external clients at a particular directory server.

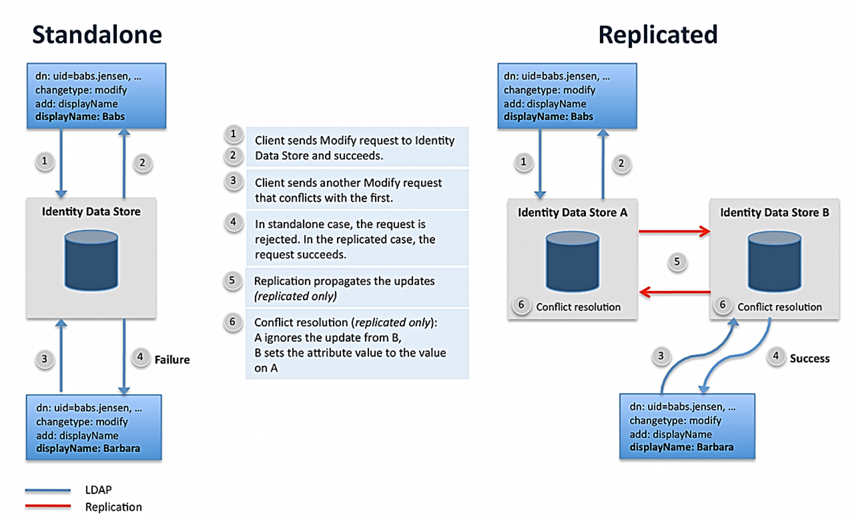
The replication protocol also sets a virtual clock that eliminates the need for synchronized time on servers. For troubleshooting purposes, however, it is still recommended to keep the system clocks synchronized.

Conflict Resolution

The eventual-consistency model employed in replication introduces a window where conflicting updates targeting the same entry may be applied at two different Directory Servers. In general, two updates to the same Directory Server are in conflict if the update that arrived later fails. Conflict resolution, when possible, corrects conflicts introduced by clients automatically. There are some exceptions, however, when manual administrative action is required. For example, adding an entry in one replica and deleting the parent of this entry on another replica simultaneously will introduce a conflict that requires manual action. In a carefully implemented deployment, the risk of introducing conflicts that require manual action can be significantly reduced or even eliminated.

The conflict resolution algorithm in the PingDirectory Server uses a mechanism that orders all updates in the replication topology. Each update in the Directory Server is assigned a unique change number. The change number is attached to each update propagated via replication and allows each Directory Server to order updates exactly the same way.

Consider the following example that results in a conflict: add a single-valued attribute with different values to an entry concurrently at two Directory Servers (shown in the figure below). It is easy to see that the second operation would fail if a client attempted to add the same attribute to the same entry at the same Directory Server. In a replicated environment, the conflict is not immediately seen if these updates are applied concurrently at two different Directory Servers. The conflict is handled only after replication propagates the updates. The Directory Servers resolve the conflict independently of the other server. On one Directory Server, the entry will be updated to reflect the correct value; on the other Directory Server, the value will stay the same. As result, each Directory Server will independently resolve the conflict the same way based on the ordering of the updates. This example is illustrated below:



1 2 Client sends Modify request to Identity Data Store and succeeds.

3 Client sends another Modify request that conflicts with the first.

4 In standalone case, the request is rejected. In the replicated case, the request succeeds.

5 Replication propagates the updates *(replicated only)*

6 Conflict resolution (*replicated only*): A ignores the update from B, B sets the attribute value to the value on A

Figure 16: Conflict Resolution Process Flow

WAN-Friendly Replication

Many multi-national corporations that have data centers in different countries must minimize latency over WAN to ensure acceptable performance for their client applications. To minimize WAN latency, the Directory Server assigns one of two roles to the replication servers: the role of standard replication transmitting updates to the other co-located replication servers; the other role, a WAN-dedicated replication server designed to send updates to other WAN-designated replication servers in other locations. This two-role system minimizes WAN traffic by pushing all replication updates onto the connected replication servers that are designated as WAN Gateway Servers. Only the designated WAN Gateway Servers can transmit the update messages to other connected WAN Gateway servers at other locations.

WAN Gateway Server

The Directory Server's replication mechanism relies on the server's location information to reduce protocol traffic on WAN links. During protocol negotiation, the replication server with the highest WAN Gateway priority (priority 1 indicates the highest priority) automatically assumes the role as the WAN Gateway Server for that particular location. The Gateway Server's main function is to route update messages from other non-gateway servers at the same location to remote WAN Gateway servers at other locations. Similarly, at the destination point, the replication server with the WAN Gateway role will receive update messages from other WAN gateway servers at other locations and push them out to all replication servers at the current location. This setup ensures that all WAN communication flows through the WAN Gateway Servers.

The figure below shows a basic connection configuration for updates. Keep in mind that all of the replication servers are fully connected to each other for monitoring or server negotiation purposes.

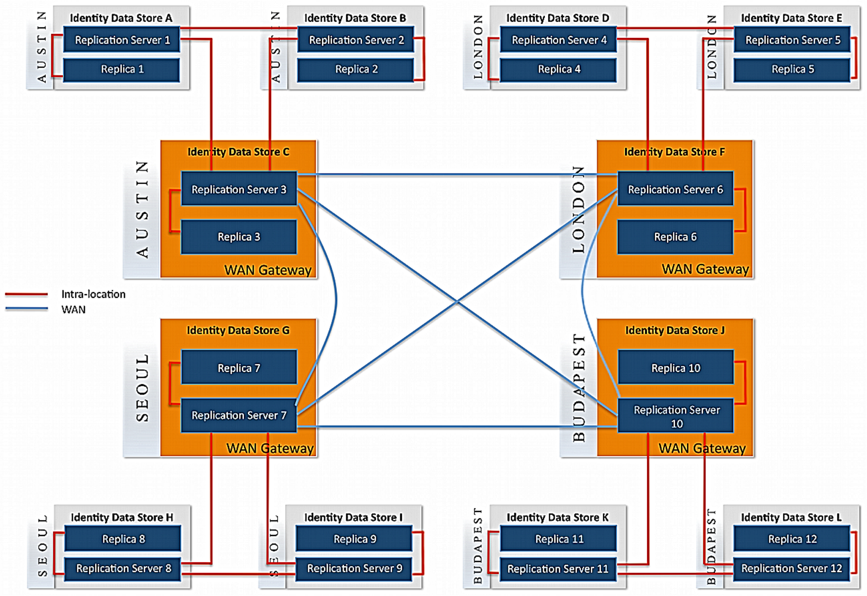


Figure 17: WAN Gateway Servers

If the WAN Gateway Server is temporarily unavailable due to a planned or unplanned downtime, the system will dynamically re-route updates to a newly designated WAN Gateway Server in the same location. The replication server with the next highest WAN Gateway priority number automatically assumes the WAN Gateway role. For deployments with entry-balancing Directory Proxy Servers, there will be one WAN Gateway Server per data set.

By default, all servers are enabled to serve as WAN Gateways and all are set to priority 5, which is simply a way to make them all equal. If necessary, the WAN Gateway priority can be changed using `dsconfig` after replication has been enabled.

WAN Message Routing

Non-gateway replication servers forward update messages from replicas to co-located replication servers only. It is the responsibility of the WAN gateway server to forward these messages to WAN gateway servers at other locations. The figure below illustrates how an update message is routed from a non-gateway server to a remote location.

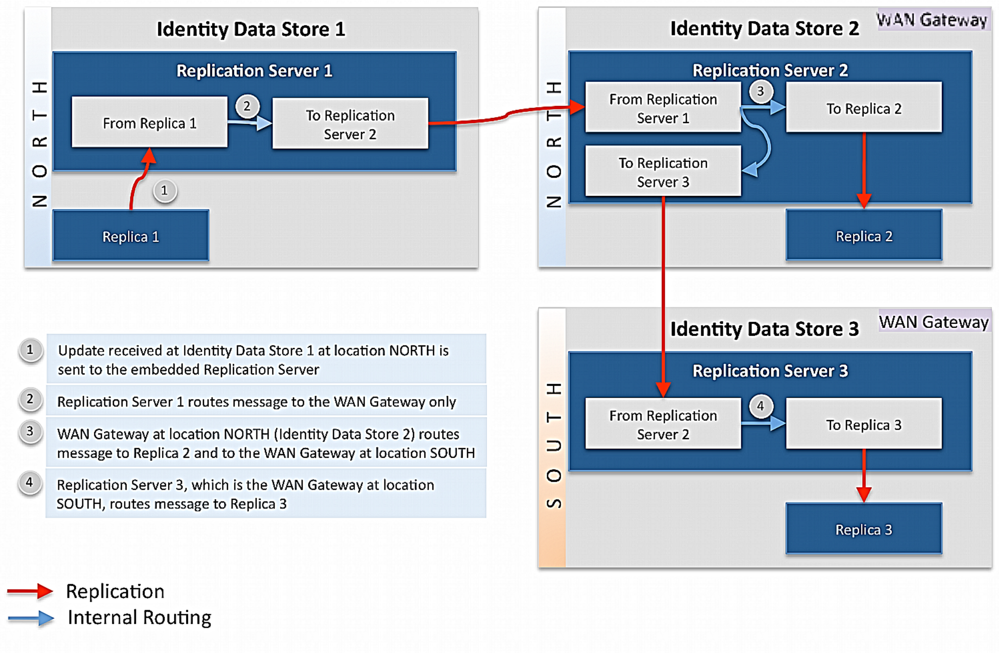

Identity Data Store 1
NORTH
Replication Server 1
From Replica 1
2
To Replication Server 2
1
Replica 1
Identity Data Store 2
WAN Gateway
NORTH
Replication Server 2
From Replication Server 1
3
To Replica 2
To Replication Server 3
Replica 2
Identity Data Store 3
WAN Gateway
SOUTH
Replication Server 3
From Replication Server 2
4
To Replica 3
Replica 3
1 Update received at Identity Data Store 1 at location NORTH is sent to the embedded Replication Server
2 Replication Server 1 routes message to the WAN Gateway only
3 WAN Gateway at location NORTH (Identity Data Store 2) routes message to Replica 2 and to the WAN Gateway at location SOUTH
4 Replication Server 3, which is the WAN Gateway at location SOUTH, routes message to Replica 3
Replication
Internal Routing

Figure 18: WAN Message Routing

WAN Gateway Server Selection

The WAN Gateway role is dynamically assigned to the most suitable server in a particular location. In most cases, the replication server with the higher WAN Gateway priority (e.g., priority 1 indicates the highest priority) assumes this role.

A replication server will not attempt to become a WAN Gateway if any of the following conditions exists:

The WAN Gateway priority is set to 0
The replication server is backlogged at server startup
The current WAN Gateway has higher priority
The WAN Gateway has not been elected yet, but higher priority replication servers are present in the location

The currently active WAN Gateway server will give up the gateway role in the following cases:

The server has started the shutdown process
The server is preparing for a scheduled maintenance cycle
The server learns about a higher priority replication server in its location

Replication servers send WAN Gateway information to other servers at regular intervals using the replication protocol. This allows the replication servers to take the appropriate action, for example, to become a WAN Gateway, if necessary without any manual administrative action.

WAN Replication in Mixed-Version Environments

WAN Gateway Role assignment is only available on PingDirectory Servers version 3.5 or later. Legacy servers (i.e., pre-3.5) will never be selected as WAN Gateways. Updates from legacy replication servers are forwarded to all replication servers and not funneled to a single WAN Gateway. Also, updates from remote WAN Gateways will be forwarded to the legacy replication servers. Both of these actions will effectively increase WAN traffic during replication.

Recovering a Replication Changelog

In the event that the replication changelog is compromised (`<server-root>/changelogDb`), possibly due to a disk or NAS failure, perform the following steps.

1. Stop the server.
2. Backup `replicationChanges` from a remote server with the following command:

```
$ bin/backup --backupDirectory /app/backups/replicationChanges \
  --backendID replicationChanges
```

3. Copy the `replicationChanges` backup from the remote server and restore it on the local host as follows:

```
$ bin/restore --backupDirectory /app/tmp/replicationChanges
```

4. Start the server.

Disaster Recovery

In the event that data is compromised across all systems and a restore is necessary, perform the following steps. These steps assume that no read or write operations are performed by any servers during this process.

1. Stop all servers.
2. Run the following command on all servers:

```
$ /bin/dsreplication cleanup-local-server
```

3. Locate the backup or exported LDIF file that represents the last working copy of the database.
4. Restore the backup or import the LDIF file on a single server. If importing an LDIF file, use the `--excludeReplication` option with the `import-ldif` command.
5. Start the restored server. The server can now receive client requests.
6. Start another server in lockdown mode with the following command:

```
$ start-server --skipPrime --lockdownMode
```

7. Enable replication from the first server to the second server.
8. Initialize the second server from the first with the following command:

```
$ bin/dsreplication initialize
```

9. Restart the second server or use the `bin/leave-lockdown-mode` command to leave the server in lockdown mode. The second server can now receive client requests.
10. Repeat steps 6 through 9 for any other servers.

Chapter

20

Managing Logging

Topics:

The PingDirectory Server supports a rich set of log publishers to monitor access, debug, and error messages that occur during normal server processing. Administrators can view the standard set of default log files as well as configure custom log publishers with pre-defined filtering with its own log rotation and retention policies.

This chapter presents the following information:

Default Directory Server Logs

The Directory Server provides a standard set of default log files to monitor the server activity. You can view this set of logs in the `PingDirectory/logs` directory. The following default log files are available on the Directory Server and are presented below.

Table 59: Directory Server Logs

Log File	Description
access	File-based Access Log that records operations processed by the Directory Server. Access log records can be used to provide information about problems during operation processing (for example, unindexed searches, failed requests, etc.), and provide information about the time required to process each operation.
change-notifications.log	Records changes to data anywhere in the server, which match one or more configured change subscriptions.
config-audit.log	Records information about changes made to the Directory Server configuration in a format that can be replayed using the `dsconfig` tool
errors	File-based Error Log. Provides information about warnings, errors, and significant events that occur during server processing.
expensive-ops	Expensive Operations Log. Disabled by default. Provides only those operations that took longer than 1000 milliseconds to complete.
failed-ops	Failed Operations Log. Provides information on all operations that failed in single-line log format.
replication	Records any replication errors and publishes them to the filesystem.
searches-returning-no-entries	Searches Returning No Entries Log. Disabled by default. Provides information only on operations that did not return any entries.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to server.out.
server.pid	Stores the server's process ID.
server.status	Stores the timestamp, a status code, and an optional message providing additional information on the server status.
setup.log	Records messages that occur during the initial configuration of a Directory Server with the `setup` tool.
tools	Directory that holds logs for long running utilities: `import-ldif`, `export-ldif`, `backup`, `restore`, `verify-index`. Current and previous copies of the log are present in the Directory Server.
update.log	Records messages that occur during a Directory Server update.

Types of Log Publishers

The PingDirectory Server provides several classes of log publishers for parsing, aggregating, and filtering information events that occur during normal processing in the server. There are three primary types of Log Publishers: access logs, debug logs, and error logs. Each type has multiple subtypes of log files based on the log server type:

Table 60: Types of Log Publishers

Log Publisher Type	Description	
Access	Provides the requests received and responses returned by the Directory Server. The information can be used to understand the operations performed by clients and to debug problems with the client applications. It can also be used for collecting usage information for performance and capacity planning purposes. There are tools described later that can analyze the access log to provide summaries of the LDAP activity and performance.	
	File-based Audit Log	Special type of access logger that provides detailed information about changes processed within the server. Disabled by default.
	JDBC-based Access Log	Stores access log information using a JDBC database connection. Disabled by default.
	File-based Access Logs	Provides a character-based stream used by TextWriter Publishers as a target for outputting log records. There are six types of file-based loggers: • **Admin Alert Access Log**. Generates administrative alerts for any operations that match the criteria for this access logger. Disabled by default. • **File-based Access Log**. Publishes access log messages to the filesystem. Enabled by default. • **Syslog-based Access Log**. Publishes access log messages to a syslogd port. Disabled by default. • **Expensive-Operations Access Log**. Publishes only those access log messages of operations that take longer than 1000 milliseconds. Disabled by default. • **Failed-Operations Access Log**. Publishes only those access log messages of operations that failed for any reason. Enabled by default. • **Successful Searches with No Entries Returned Log**. Publishes only those access log messages of search operations that failed to return any entries. Disabled by default.
Debug	Provides information about warnings, errors, or significant events that occur within the server.	
	Debug ACI Logger	Stores debug information on ACI evaluation for any request operations against the server
	File-based Error Logs	There are two types of File-based Error Logs: • **Error log**. Publishes error messages to the filesystem. Enabled by default. • **Replication log**. Publishes replication error messages to the filesystem. Enabled by default.
	JDBC-based Error Logs	Stores error log information using a JDBC database connection. Disabled by default.
	Syslog-based Error Logs	Publishes error messages to a syslogd port.

Viewing the List of Log Publishers

You can quickly view the list of log publishers on the Directory Server using the `dsconfig` tool.

Note: Initially, the JDBC, syslog, and Admin Alert log publishers must specifically be configured using `dsconfig` before they appear in the list of log publishers. Procedures to configure these types of log publishers appear later in this chapter.

To View the List of Log Publishers

- Use `dsconfig` to view the log publishers.

```
$ bin/dsconfig list-log-publishers
```

```
Log Publisher                                : Type               : enabled
---------------------------------------------:--------------------:--------
Debug ACI Logger                             : debug-access       : false
Expensive Operations Access Logger           : file-based-access  : false
Failed Operations Access Logger              : file-based-access  : true
File-Based Access Logger                     : file-based-access  : true
File-Based Audit Logger                      : file-based-audit   : false
File-Based Debug Logger                      : file-based-debug   : false
File-Based Error Logger                      : file-based-error   : true
Replication Repair Logger                    : file-based-error   : true
Successful Searches with No Entries Returned : file-based-access  : false
```

Enabling or Disabling a Default Log Publisher

You can enable or disable any log publisher available on the Directory Server using the `dsconfig` tool. By default, the following loggers are disabled and should be enabled only when troubleshooting an issue on the server:

Expensive Operations Access Logger
File-Based Audit Logger
File-Based Debug Logger
Successful Searches with No Entries Returned

To Enable a Default Access Log

- Use `dsconfig` to enable an Access Log Publisher. In this example, enable the Expensive-Ops log, which will record only those access log messages that take 1000 milliseconds or longer.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Expensive Operations Access Logger" --set enabled:true
```

Managing Access and Error Log Publishers

The Access Log records every request received and response returned by the Directory Server. The Access Log stores the IDs for the client connection, operation, the LDAP message involved with each client request, and the server response. The information can be used to debug any problems with a client application by correlating the numeric operation identifier to the client request or response.

The Directory Server supports multiple classes of access log publishers depending on your logging requirements. The following types of access log publishers are available on the system:

- **File-Based Access Log Publishers**. Provides a character-based TextWriter stream for outputting log records. There are three subclasses of TextWriter access logs:
  - **File-Based Access Logs**. Enabled by default. The File-based Access Log publishes access messages to the filesystem as `<server-root>/logs/access`. The Failed-Operations Log, Expensive-Operations Log, and the Searches with No Entries Returned Log are specialized types of the File-Based Access Log and shows only specific information necessary for troubleshooting purposes.

  - **Admin-Alert Access Logs**. Disabled by default. The Admin-Alert Access Log is specialized type of logger that automatically generates administrative alerts for any operations that match a criteria for this access log publisher.
  - **Syslog-Based Access Logs**. Disabled by default. The Syslog Access Log publishes access messages to a syslogd port.
- **File-Based Audit Logs**. Disabled by default. The Audit Log provides detailed information about modifications (writes) processed within the Directory Server. The File-based Audit Log publishes access messages to the filesystem as `<server-root>/logs/audit`.
- **JDBC-Based Access Logs**. Disabled by default. The JDBC-based Access Log provides information using a JDBC database connection.
- **JDBC-Based Error Logs**. Disabled by default. The JDBC-based Error Log provides information using a JDBC database connection.

Managing File-Based Access Log Publishers

The PingDirectory Server supports a flexible and configurable Access Logging system that provides a full set of customized components to meet your system's logging requirements. The default Access Log can be configured to write information to a log file with two records per operation, one for the request received and one for the response returned. It can also be configured to write one message per operation or configured to record information about a subset of operations processed by the server. In addition to modifying existing default log files, you can create custom log publishers to monitor specific properties or connection criteria. For more information, see *Creating New Log Publishers*.

The Directory Server can be configured to use multiple access log publishers writing logs to different files using different configurations. This approach makes it possible to have fine-grained logging for various purposes (for example, a log that contains only failed operations, a log that contains only operations requested by root users, or a log that contains only operations that took longer than 20ms to complete).

The Directory Server provides an additional mechanism to filter access logs to record only a subset of messages of one or more types. The access log filtering mechanism uses the operation criteria (connection, request, result, search-entry, search-reference) to determine whether a given message should be logged based on information associated with the client connection as well as information in the operation request/response messages. For more information, see *Configuring Filtered Logging*.

Access Log Format

The Access Log has a standard format that lists various elements identifying the connection and operation occurring within the Directory Server. By default, each operation generates one access log message.

The Access Log displays the following common properties:

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Connection Type**. Displays the connection type requested by the client and the response by the server. Examples include the following:

  CONNECT
  BIND REQUEST/RESULT
  UNBIND REQUEST
  DISCONNECT
  SEARCH REQUEST/RESULT
  MODIFY REQUEST/RESPONSE
  others include: ABANDON, ADD, COMPARE, DELETE, EXTENDED OPERATION, MODIFY, MODIFY DN
- **Connection ID**. Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation. The connection ID is unique for a span of time on a single server. Values of the

connection ID will be re-used when the server restarts or when it has had enough connections to cause the identifier to wrap back to zero.

- **Operation ID**. Numeric identifier, starting incrementally with 0, that identifies the operation. The operation ID is unique for a span of time on a single server. Values of the operation ID will be re-used when the server restarts or when it has serviced enough operations to cause the identifier to wrap back to zero.
- **Result Code**. LDAP result code that determines the success or failure of the operation result. Result messages include a result element that indicates whether the operation was successful or if failed, the general category for the failure, and an etime element that indicates the length of time in milliseconds that the server spent processing the operation.

  The Directory Server provides a useful tool `<server-root>/bin/ldap-result-code` (UNIX, Linux) or `<server-root>\bat\ldap-result-code` (Windows), that displays all of the result codes used in the system. You can use the utility if you are not sure what a result code means. For example, use the following:

  - `ldap-result-code` --list displays all of the defined result codes in the Directory Server.
  - `ldap-result-code` --int-value 16654 displays the name of the result code with a numeric value of 16654.
  - `ldap-result-code` --search operation displays a list of all result codes whose name includes the substring "operation".
- **Elapsed Time**. Displays the elapsed time (milliseconds) during which the operation completed its processing.
- **Message ID**. Numeric identifier, starting incrementally with 1, which identifies the LDAP message used to request the operation.

Access Log Example

The following example shows output from the Access Log in `<server-root>/logs/access`:

```
[01/Jun/2011:14:48:17 -0500] CONNECT conn=0 from="10.8.1.243" to="10.8.1.243"
 protocol="LDAP"
[01/Jun/2011:14:48:17 -0500] BIND REQUEST conn=0 op=0 msgID=1 version="3"
 dn="cn=Directory Manager"
  authType="SIMPLE"
[01/Jun/2011:14:48:17 -0500] BIND RESULT conn=0 op=0 msgID=1 resultCode=0
 etime=26.357
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
[01/Jun/2011:14:48:17 -0500] UNBIND REQUEST conn=0 op=1 msgID=2
[01/Jun/2011:14:48:17 -0500] DISCONNECT conn=0 reason="Client Unbind"
... (more output) ...
```

Modifying the Access Log Using dsconfig Interactive Mode

The File-Based Access Log can be modified to include or exclude all log messages of a given type using the `dsconfig` tool in interactive or non-interactive mode.

To Modify the Access Log Using dsconfig Interactive Mode

1. Use `dsconfig` in interactive mode to modify the access log properties.

   ```
   $ bin/dsconfig
   ```
2. Follow the prompts to specify the LDAP connection parameters for host name or IP address, connection type (LDAP, SSL, or StartTLS), port number, bind DN and password.
3. On the Directory Server main menu, type the number corresponding to the Log Publisher.
4. On the **Log Publisher Management** menu, enter the option to view and edit an existing log publisher.
5. On the **Log Publisher** menu, type the number corresponding to File-based Access Logger.
6. On the **File-Based Access Log Publisher** menu, type the number corresponding to the property that you want to change, and then follow the prompts.
7. Type `f` to apply the changes.

Modifying the Access Log Using dsconfig Non-Interactive Mode

You can use the `dsconfig` tool in non-interactive mode to quickly modify a log publisher property on the command line or in a script. For information on each property, see the *Directory Server Configuration Reference Guide*, which is an HTML document listing the various properties for each Directory Server component.

To Modify the Access Log Using dsconfig Non-Interactive Mode

- Use `dsconfig` with the --no-prompt option with the properties that you want to modify or set for your access log. In this example, enable the properties to include the instance name and startup ID.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Access
 Logger" \
  --set include-instance-name:true --set include-startup-id:true
```

Modifying the Maximum Length of Log Message Strings

By default, the Directory Server sets the maximum length of log message strings to 2000 characters. This value is configurable for any access log publisher, except the Syslog publisher, which is set to 500 characters. You can change the maximum length of log message strings by setting the `max-string-length` configuration property. If any string has more than the configured number of characters, then that string will be truncated and a placeholder will be appended to indicate the number of remaining characters in the original string.

To Modify the Maximum Length of Log Message Strings

- Use `dsconfig` to set the `max-string-length` property for an access log. The following command configures the "File-based Access Logger" to include the instance name and the maximum length of the log message strings to 5000 characters.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set include-instance-name:true \
  --set max-string-length:5000
```

Generating Access Logs Summaries

The Directory Server provides a convenience tool, `summarize-access-log`, that generates a summary of one or more file-based access logs. The summary provides analytical metric information that could be useful for administrators. The following metrics are provided in each summary:

- Total time span covered by the log files.
- Number of connections established and the average rate of new connections per second.
- IP addresses of up to the top 20 of the clients that most frequently connect to the server, the number of connections by each address, and the percentage of connections of each.
- Total number of operations processed, the number of operations of each type, the percentage of each type out of the total number, and the average rate per second for each type of operation.
- Average processing time for all operations and for each type of operation.
- Histogram of the processing times for each type of operation.
- Up to the 20 most common result codes for each type of operation, the number of operations of that type with that result code, and the percentage of operations of that type with that result code.
- Number of unindexed searches processed by the server.
- Breakdown of the scopes used for search operations with the number of percentage of searches with each scope.
- Breakdown of the most common entry counts for search operations with the number and percentage of searches that returned that number of entries.
- Breakdown of the most commonly used filter types for searches with a scope other than "base" (that is, those searches for which the server will use es when processing the filter). These filters will be represented in a generic manner so that any distinct assertion values or substring assertion elements will be replaced with question marks

and attribute names in all lowercase characters (for example, `(givenName=John)` would be represented as `(givenName=?)`).

To Generate an Access Log Summary

- Use the `bin/summarize-access-log` with path to one or more access log files.

```
$ bin/summarize-access-log /path/to/logs/access
```

```
Examining access log /path/to/logs/access Examined 500 lines in 1 file
 covering a total duration of 1 day, 22 hours, 57 minutes, 31 seconds

Total connections established: 69 (0.000/second)
Total disconnects: 69 (0.000/second)

Most common client addresses:
127.0.0.1: 61 (88.406)
10.8.1.209: 8 (11.594)

Total operations examined: 181 ...
(metric for each operation examined) ...

Average operation processing duration: 22.727ms
Average add operation processing duration: 226.600ms
Average bind operation processing duration: 5.721ms
Average delete operation processing duration: 77.692ms
Average modify operation processing duration: 35.530ms
Average search operation processing duration: 4.017ms

Count of add operations by processing time:
... (histogram for add operations) ...

Count of bind operations by processing time:
... (histogram for bind operations) ...

Count of delete operations by processing time:
... (histogram for delete operations) ...

Count of modify operations by processing time:
... (histogram for modify operations) ...

Count of search operations by processing time:
... (histogram for search operations) ...

Most common add operation result codes:
success: 11 (84.615%)
entry already exists: 2 (15.385%)

Most common bind operation result codes:
success: 4 (50.000%)
invalid credentials: 4 (50.000%)

Most common delete operation result codes:
success: 1 (100.000%)

Most common modify operation result codes:
success: 9 (69.231%)
no such object: 4 (30.769%)

Most common search operation result codes:
success: 133 (91.724%)
no such object: 12 (8.276%)
```

```
Number of unindexed searches: 0

Most common search scopes:
BASE: 114 (78.621%)
SUB: 16 (11.034%)
ONE: 15 (10.345%)

Most common search entry counts:
1: 119 (82.069%)
0: 17 (11.724%)
2: 5 (3.448%)
10: 4 (2.759%)

Most common generic filters for searches with a non-base scope:
(objectclass=?): 19 (61.290%)
(ds-backend-id=?): 12 (38.710%)
```

About Log Compression

The Directory Server supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The `summarize-access-log` tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting purposes, administrators using compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the `compression-mechanism` property to have the value of "gzip" when creating a new logger.

About Log Signing

The Directory Server supports the ability to cryptographically sign a log to ensure that it has not been modified in any way. For example, financial institutions require audit logs for all transactions to check for correctness. Tamper-proof files are therefore needed to ensure that these transactions can be propertly validated and ensure that they have not been modified by any third-party entity or internally by unscrupulous employees. You can use the `dsconfig` tool to enable the `sign-log` property on a Log Publisher to turn on cryptographic signing.

When enabling signing for a logger that already exists and was enabled without signing, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the `validate-file-signature` tool provided in the `bin` directory of the server (or the `bat` directory for Windows systems).

Once you have enabled this property, you must disable and then re-enable the Log Publisher for the changes to take effect.

About Encrypting Log Files

The Directory Server supports the ability to encrypt log files as they are written. The `encrypt-log` configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an `.encrypted` extension (and if both encryption and compression are enabled, the extension will be `.gz.encrypted`). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The `encrypt-file` tool (or custom code, using the LDAP SDK's `com.unboundid.util.PassphraseEncryptedInputStream`) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the `encryption-settings create` command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the `encryption-settings-definition-id` property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the `rotate-log` tool to force the file to be rotated before attempting to examine it.

To Configure Log Signing

1. Use `dsconfig` to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set enabled:false
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set enabled:true
```

To Validate a Signed File

The Directory Server provides a tool, `validate-file-signature`, that checks if a file has not been tampered with in any way.

- Run the `validate-file-signature` tool to check if a signed file has been tampered with. For this example, assume that the `sign-log` property was enabled for the File-Based Audit Log Publisher.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

Note: If any validations errors occur, you will see a message similar to the one as follows:

```
One or more signature validation errors were encountered
while validating the contents of file 'logs/audit':
* The end of the input stream was encountered without
  encountering the end of an active signature block.
  The contents of this signed block cannot be trusted
  because the signature cannot be verified
```

To Configure Log File Encryption

1. Use `dsconfig` to enable encryption for a Log Publisher. In this example, the File-based Access Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted Access"
 \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-definition-
id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...Done
Writing decrypted data to file '/ds/PingDirectory/decrypted-access' using a
key generated from encryption settings definition
 '332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'
Successfully wrote 123,456,789 bytes of decrypted data
```

Creating New Log Publishers

The PingDirectory Server provides customization options to help you create your own log publishers with the `dsconfig` command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see Configuring Log Rotation and Configuring Log Retention.

To Create a New Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

Note: To configure compression on the logger, add the option to the previous command:

```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. If needed, view log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. On the main menu, select the option to configure the log publisher.
4. On the **Log Publisher** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the Directory Server root. For example, `logs/disconnect.log`.
9. Select the rotation policy to use for this log publisher.
10. Select the retention policy to use for this log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Type `f` to apply the changes.

Configuring Log Rotation

The Directory Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

To Configure the Log Rotation Policy

- Use `dsconfig` to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring Log Rotation Listeners

The Directory Server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed. Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true
```

The path specified by the `copy-to-directory` property must exist, and the filesystem containing that directory must have enough space to hold all of the log files that will be written there. The server will automatically monitor free disk space on the target filesystem and will generate administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the `summarize-access-log` tool on a recently-rotated log file and writes its output to a file in a specified location. This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the `summarize-access-log` tool, including the `file-based-access` and `operation-timing-access` logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of `.summary`. If the `output-directory` property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically be deleted. Though files are generally small in comparison to the log files themselves, make sure that there is enough space available in the specified storage directory. The server automatically monitors free disk space on the filesystem to which the summary files are written.

Configuring Log Retention

The Directory Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Directory Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Custom Retention Policy**. Create a new retention policy that meets your Directory Server’s requirements. This will require developing custom code to implement the desired log retention policy.

- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

- Use `dsconfig` to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configuring Filtered Logging

The PingDirectory Server provides a mechanism to filter access log messages based on specific criteria. The filtered log can then be used with a custom log publisher to create and to generate your own custom logs. Adding new filtered logs and associate publishers does not change the behavior of any existing logs. For example, adding a new log that only contains operations that were unsuccessful does not result in those operations being removed from the default access log.

The following example shows how to create a set of criteria that matches any operation that did not complete successfully. It then explains how to create a custom access log publisher that logs only operations matching that criteria. Note that this log does not include messages for connects or disconnects, and only a single message is logged per operation. This message contains both the request and result details.

To run log filtering based on any operation result (for example, result code, processing time, and response controls), turn off request logging and set the `include-request-details-in-result-messages` property to TRUE. Since filtering based on the results of an operation cannot be done until the operation completes, the server has no idea whether to log the request. Therefore, it might log request messages but not log any result messages. Instead, if you can only log result messages and include request details in the result messages, then only messages for operations that match the result criteria are logged. All pertinent information about the corresponding requests are included.

To Configure a Filtered Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create a result criteria object set to failure-result-codes, a predefined set of result codes that indicate when an operation did not complete successfully.

```
$ bin/dsconfig create-result-criteria --type simple \
  --criteria-name "Failed Operations" --set result-code-criteria:failure-
result-codes
```

2. Use `dsconfig` to create the corresponding log publisher that uses the result criteria. The log rotation and retention policies are also set with this command.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access \
  --publisher-name "Filtered Failed Operations" \
  --set enabled:true \
  --set log-connects:false \
  --set log-disconnects:false \
  --set log-requests:false \
  --set "result-criteria:Failed Operations" \
  --set log-file:logs/failed-ops.log \
  --set include-request-details-in-result-messages:true \
  --set "rotation-policy:7 Days Time Limit Rotation Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

3. View the `failed-ops.log` in the `logs` directory. Verify that only information about failed operations is written to it.

Managing Admin Alert Access Logs

Admin Alert Access Logs are a specialized form of filtered log that automatically generates an administrative alert when criteria configured for the log publisher matches those messages in the access log.

About Access Log Criteria

Configuring an Admin Alert Access Log requires that you configure the criteria for the access log messages. Each criteria can be either a Simple or an Aggregate type. The Simple type uses the set of properties for the client connection, operation request, and the contents of any operation-specific requests or results. The Aggregate type provides criteria that contains Boolean combination of other operation-specific criteria objects. For more information, see the *Ping Identity Directory Server Configuration Reference*.

The criteria can be one or more of the following:

- **Connection Criteria**. Defines sets of criteria for grouping and describing client connections based on a number of properties, including protocol, client address, connection security, and authentication state.
- **Request Criteria**. Defines sets of criteria for grouping and describing operation requests based on a number of properties, including properties for the associated client connection, the type of operation, targeted entry, request controls, target attributes, and other operation-specific terms.
- **Result Criteria**. Defines sets of criteria for grouping and describing operation results based on a number of properties, including the associated client connection and operation request, the result code, response controls, privileges missing or used, and other operation-specific terms.
- **Search Entry Criteria**. Defines sets of criteria for grouping and describing search result entries based on a number of properties, including the associated client connection and operation request, the entry location and contents, and included controls.
- **Search Reference Criteria**. Defines sets of criteria for grouping and describing search result references based on a number of properties, including the associated client connection and operation request, reference contents, and included controls.

Configuring an Admin Alert Access Log Publisher

Prior to configuring an Admin Alert Access Log, you must establish an administrative alert handler in your system. For more information, see *Working with Administrative Alert Handlers*.

To Configure an Admin Alert Access Log Publisher

1. Use `dsconfig` to create a criteria object for the Admin Alert Access Log. For this example, we want to log only write operations that target user entries. The following command matches any of the specified operations whose target entry matches the filter “`(objectClass=person)`”.

   If you are using the `dsconfig` tool in interactive mode, the menu items for the criteria operations are located in the Standard objects menu.

   ```
   $ bin/dsconfig create-request-criteria --type simple \
     --criteria-name “User Updates” \
     --set operation-type:add \
     --set operation-type:delete \
     --set operation-type:modify \
     --set operation-type:modify-dn \
     --set “any-included-target-entry-filter:(objectClass=person)”
   ```

2. Use `dsconfig` to create a log publisher of type `admin-alert-access`.

   ```
   $ bin/dsconfig create-log-publisher \
     --publisher-name ”User Updates Admin Alert Access Log” \
     --type admin-alert-access \
     --set “request-criteria:User Updates” \
     --set include-request-details-in-result-messages:true \
     --set enabled:true
   ```

Managing Syslog-Based Access Log Publishers

The Directory Server supports access logging using the syslog protocol that has been part of the Berkeley Software Distribution (BSD) operating systems for many years. Syslog provides a flexible, albeit simple, means to generate, store and transfer log messages that is supported on most UNIX and Linux operating systems.

The quasi-standard syslog message format cannot exceed 1 kbytes and has three important parts:

- **PRI**. Specifies the message priority based on its facility and severity. The message facility is a numeric identifier that specifies the type of log messages, such as kernel messages, mail system messages, etc. The severity is a numeric identifier that specifies the severity level of the operation that is being reported. Together, the facility and the severity determine the priority of the log message indicated by angled brackets and 1-3 digit priority number. For example, "<0>", "<13>", "<103>" are valid representations of the PRI.
- **Timestamp and Host Name**. The timestamp displays the current date and time of the log. The host name or IP address displays the source of the log.
- **Message**. Displays the actual log message.

Administrators can configure syslog to handle log messages using log priorities that are based on the message's facility and severity. This feature allows users to configure the logging system in such a way that messages with high severities can be sent to a centralized repository, while lower severity messages can be stored locally on a server.

Note: Since the numeric values of the severity and facility are operating system-dependent, the central repository must only include syslog messages from compatible OS types, otherwise the meanings of the PRI field is ambiguous.

Before You Begin

You will need to identify the host name and port to which you want to connect. Because the Syslog Protocol uses User Datagram Protocol (UDP) packets, we highly recommend that you use `localhost` and utilize some additional logging tools, such as `syslog-ng`. UDP is unreliable and unsecure means to transfer data packets between hosts.

Default Access Log Severity Level

All messages are logged at the syslog severity level of 6, which is `Informational: informational` messages. Note that this value is not standard across different types of UNIX or Linux systems. Please consult your particular operating system.

Syslog Facility Properties

When using syslog, specify a facility for the access log messages. As an advanced property, you can select a number that corresponds to the facility you wish to use. The default value for the `syslog-facility` property is 1 (one) for user-level messages. Note that these values are not standard across different types of UNIX or Linux systems. Please consult your particular operating system documentation for properties specific to that system.

Table 61: Syslog Facility Properties

Facility	Description
0	kernel messages
1	user-level messages (default)
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem

Facility	Description
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local use 0
17	local use 1
18	local use 2
19	local use 3
20	local use 4
21	local use 5
22	local use 6
23	local use 7

Queue-Size Property

The maximum number of log records that can be stored in the asynchronous queue is determined by the `queue-size` property. The default queue size set is 10000, which means that the server will continuously flush messages from the queue to the log. The server does not wait for the queue to fill up before flushing to the log. Therefore, lowering this value can impact performance.

Configuring a Syslog-Based Access Log Publisher

You can configure a Syslog-based Access Log Publisher using the `dsconfig` tool. We recommend that you use syslog locally on localhost and use `syslog-ng` to send the syslog messages to remote syslog servers.

Because syslog implementations differ by vendor, please review your particular vendor's syslog configuration.

To Configure a Syslog-Based Access Log Publisher

- Use `dsconfig` to create a log publisher of type syslog-based-access.

  If you are using the `dsconfig` tool in interactive mode, the menu item for Syslog Facility is an advanced property, which can be exposed by typing `a` (for "show advanced properties") on the Syslog-Based Access Log Publisher menu.

```
$ bin/dsconfig create-log-publisher \
--publisher-name "syslog-access" \
--type syslog-based-access \
--set syslog-facility:4 \
--set enabled:true
```

Managing the File-Based Audit Log Publishers

The Directory Server provides an audit log, a specialized version of the access log, for troubleshooting problems that may occur in the course of processing. The log records all changes to the data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

Audit Log Format

The audit log uses standard LDIF format, so that administrators can quickly analyze what changes occurred to the data. The audit log begins logging when enabled and should be used to debug any issues that may have occurred. Some common properties are the following:

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Connection ID**. Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation.
- **Operation ID**. Numeric identifier, starting incrementally with 0, that identifies the operation.
- **Modifiers Name**. Displays the DN of the user who made the change.
- **Update Time**. Records the `modifyTimestamp` operational attribute.

Audit Log Example

The following example shows output from the Audit Log in the `<server-root>/ logs/audit`. The first entry shows when the audit log was enabled. The second entry show changes made to a user entry.

```
# 05/Jun/2011:10:29:04 -0500; conn=0; op=55
dn: cn=File-Based Audit Logger,cn=Loggers,cn=config
changetype: modify
replace: ds-cfg-enabled
ds-cfg-enabled: true
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z

# 05/Jun/2011:10:31:20 -0500; conn=2; op=1
dn: uid=user.996,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 115 426 4748
-
replace: homePhone
homePhone: +1 407 383 4949
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
```

Enabling the File-Based Audit Log Publisher

You can enable the File-Based Audit Log Publisher using the `dsconfig` tool. The audit log can impact the Directory Server's write performance, so enable it only when troubleshooting any issues.

To Enable the File-Based Audit Log Publisher

- Use `dsconfig` to enable the File-Based Audit Log Publisher. For this example, the instance name and startup ID is also enabled in the audit log.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true \
  --set include-instance-name:true \
  --set include-startup-id:true
```

Obscuring Values in the Audit Log

You can obscure the values of specific attributes in the audit log using the `obscure-attribute` property. Each value of an obscured attribute is replaced in the audit log with a string of the form "***** OBSCURED VALUE *****". By default, attributes are not obscured, because the values of password attributes appear in hashed form rather than in the clear.

Managing the JDBC Access Log Publishers

The Directory Server supports the Java Database Connectivity (JDBCTM) API, which allows access to SQL data stores by means of its JDBC drivers. The JDBC 4.0 API, part of the Java SDK, provides a seamless method to interface with various database types in heterogeneous environments.

By easily connecting to a database, the Directory Server can be configured to implement a centralized logging system with different databases. Centralized logging simplifies log correlation and analysis tasks and provides security by storing data in a single repository. Some disadvantages of centralized logging are that data flow asymmetries may complicate synchronization or network provisioning and may unduly burden the central repository with possibly heavy loads.

Before You Begin

Before configuring the JDBC Access Log Publisher, you need to carry out two essential steps to set up the database.

- Install the database drivers in the Directory Server `lib` directory.
- Define the log mapping tables needed to map access log elements to the database column data. Only those elements in the log mapping table gets logged by the JDBC Log Publisher.

The following sections provide more information about these tasks.

Configuring the JDBC Drivers

The Directory Server supports a number of JDBC drivers available in the market. We highly recommend using the JDBC 4 drivers supported in the Java platform. For example, for Oracle databases, you need to use the `ojdbc.jar` driver for Java and any associated JAR files (National Language Support jars, and others) required to connect with the particular database. The following databases are supported:

DB2
MySQL
Oracle Call Interface (OCI)
Oracle Thin
PostgreSQL
SQL Server

To Configure the JDBC Driver

- Obtain the JAR file(s) for your particular database, and copy it into the `<server-root>/lib` directory.

Configuring the Log Field Mapping Tables

The log field mapping table associates access log fields to the database column names. You can configure the log field mapping table using the `dsconfig` tool, which then generates a DDL file that you can import into your database. The DDL file is generated when you create the JDBC Log Publisher.

To uniquely identify a log record, we recommend always mapping the following fields: `timestamp`, `startupid`, `message-type`, `connection-id`, `operation-type`, `instance-name`.

The table name is not part of this mapping.

The Directory Server also provides three options that you can quickly select for creating a log field mapping table:

- **Complete JDBC Access Log Field Mappings**. Maps all 52 object properties.
- **Complete JDBC Error Log Field Mappings**. Maps all 8 object properties.
- **Simple JDBC Access Log Field Mappings**. Maps a common set of object properties.
- **Custom JDBC Access Log Field Mappings**. Create a custom set of JDBC log field mappings.
- **Custom JDBC Error Log Field Mappings**. Create a custom set of JDBC error log field mappings.

To Configure the Log Field Mapping Tables

1. Use `dsconfig` to create a log field mapping table. On the main menu, type `o` to change to the Standard Object menu, and type the number corresponding to Log Field Mapping.
2. On the **Log Field Mapping management** menu, enter the option to create a new Log Field Mapping.
3. On the **Log Field Mapping template** menu, enter the option to select a complete JDBC Access Log Field mapping to use as a template for your new field mapping.
4. Next, enter a name for the new field mapping. In this example, type `my-jdbc-test`.
5. On the **Access Log Field Mapping Properties** menu, select a property for which you want to change the value. Any property that is undefined will not be logged by the JDBC Access Log Publisher. When complete, type `f` to save and apply the changes.
6. On the **Log Field Mapping Management** menu, type `q` to exit the menu.
7. View the existing Log Mappings on the system.

```
$ bin/dsconfig list-log-field-mappings
```

```

Log Field Mapping                      : Type
----------------------------------------:-------
Complete JDBC Access Log Field Mappings : access
Complete JDBC Error Log Field Mappings  : error
my-jdbc-test                            : access
Simple JDBC Access Log Field Mappings   : access
```

Configuring the JDBC Access Log Publisher using dsconfig Interactive Mode

After setting up the drivers and the log mapping table, use the `dsconfig` utility to configure the JDBC Access Log Publisher on the Directory Server. The following example uses `dsconfig` interactive mode to illustrate the steps required to configure the log publisher and the external database server.

To Configure the JDBC Access Log Publisher

1. Copy the database JAR files to the `<server-root>/lib` directory, and then restart the Directory Server.
2. Launch the `dsconfig` tool in interactive command-line mode.

```
$ bin/dsconfig
```

3. Next, type the connection parameters to bind to the Directory Server. Enter the host name or IP address, type of LDAP connection (LDAP, SSL, or StartTLS) that you are using on the Directory Server, the LDAP listener port number, the user bind DN, and the bind DN password.
4. On the main menu, type the number corresponding to Log Publisher.
5. On the **Log Publisher management** menu, enter the option to create a new Log Publisher.
6. On the **Log Publisher template** menu, type n to create a new Log Publisher.
7. On the **Log Publisher Type** menu, enter the option to create a new JDBC-Based Access Log Publisher.
8. Type a name for the JDBC Access Log Publisher.
9. On the **Enabled Property** menu, enter the option to enable the log publisher.
10. On the **Server Property** menu, enter the option to create a new JDBC External Server.
11. Next, type the name for the JDBC External Server. This is a symbolic name used to represent the DBMS.
12. On the **JDBC Driver Type Property** menu, type the number corresponding to the type of JDBC database driver type.
13. Next, type a name for the `database-name` property. This is the DBMS database name. The database name must contain the table referred to in the generated DDL.
14. Next, type the host name or IP address (server-host-name) of the external server.
15. Type the server listener port. In this example, type `1541`.
16. Review the properties for the external server, and the type `f` to apply the changes.
17. If you need to supply your own JDBC URL, type a for advanced properties to open the `jdbc-driver-url` property and supply the appropriate URL. The example below shows how to access an Oracle Thin Client connection using a SID instead of a Service.

```
>>>> Configure the properties of the JDBC External Server

Property Value(s)
-----------------------------------------------------
1) description -
2) jdbc-driver-type oraclethin
3) jdbc-driver-url jdbc:oracle:thin@myhost:1541:my_SID
4) database-name jdbc-test
5) server-host-name localhost
6) server-port 1541
7) user-name -
8) password -

?) help
f) finish - create the new JDBC External Server
a) hide advanced properties of the JDBC External Server
d) display the equivalent dsconfig arguments to create this object
b) back
q) quit

Enter choice [b]: f
```

```
JDBC External Server was created successfully
```

18. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `<server-root>/logs/ddls/<name-of-logger>.sql` file. Import the DDL file to your database.

Configuring the JDBC Access Log Publisher Using dsconfig Non-Interactive Mode

The following example uses `dsconfig` non-interactive mode to illustrate the steps to configure the log publisher and the external database server presented in the previous section.

To Configure the JDBC Access Log Publisher in Non-Interactive Mode

1. Use `dsconfig` with the --no-prompt option to create the JDBC external server.

   ```
   $ bin/dsconfig --no-prompt create-external-server \
     --server-name jdbc-external \ --type jdbc \
     --set jdbc-driver-type:oraclethin \
     --set database-name:ubid_access_log \
     --set server-host-name:localhost --set server-port:1541
   ```
2. Use `dsconfig` to create the log publisher.

   ```
   $ bin/dsconfig --no-prompt create-log-publisher \
     --publisher-name jdbc-test \
     --type jdbc-based-access \
     --set enabled:true \
     --set server:jdbc-external \
     --set "log-field-mapping:Simple JDBC Access Log Field Mappings"
   ```
3. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `&lt;server-root>/logs/ddls/&lt;name- of-logger>.sql` file. Import the DDL file to your database.

   The procedure to configure the JDBC-Based Error Log Publisher is similar to creating a JDBC-Based Access Log Publisher. You can run the previous `dsconfig` command with the --type jdbc-based-error as follows:

   ```
   $ bin/dsconfig --no-prompt create-log-publisher \
   --publisher-name jdbc-error-test \
   --type jdbc-based-error \
   --set enabled:true \
   --set server:jdbc-external \
   --set "log-field-mapping:Simple JDBC Access Log Field Mappings"
   ```

Managing the File-Based Error Log Publisher

The Error Log reports errors, warnings, and informational messages about events that occur during the course of the Directory Server’s operation. Each entry in the error log records the following properties (some are disabled by default and must be enabled):

- **Time Stamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Category**. Specifies the message category that is loosely based on the server components.
- **Severity**. Specifies the message severity of the event, which defines the importance of the message in terms of major errors that need to be quickly addressed. The default severity levels are: `fatal-error`, `notice`, `severe-error`, `severe-warning`.
- **Message ID**. Specifies the numeric identifier of the message.
- **Message**. Stores the error, warning, or informational message.

Error Log Example

The following example displays the error log for the PingDataMetrics Server. The log is enabled by default and is accessible in the `<server-root>/logs/errors` file.

```
[21/Oct/2012:05:15:23.048 -0500] category=RUNTIME_INFORMATION severity=NOTICE
msgID=20381715 msg="JVM Arguments: '-Xmx8g', '-Xms8g', '-XX:MaxNewSize=1g',
'-XX:NewSize=1g', '-XX:+UseConcMarkSweepGC', '-XX:+CMSConcurrentMTEnabled',
'-XX:+CMSParallelRemarkEnabled', '-XX:+CMSParallelSurvivorRemarkEnabled',
'-XX:+CMSScavengeBeforeRemark', '-XX:RefDiscoveryPolicy=1',
'-XX:ParallelCMSThreads=4', '-XX:CMSMaxAbortablePrecleanTime=3600000',
'-XX:CMSInitiatingOccupancyFraction=80', '-XX:+UseParNewGC', '-XX:
+UseMembar',
```

```
'-XX:+UseBiasedLocking', '-XX:+UseLargePages', '-XX:+UseCompressedOops',
'-XX:PermSize=128M', '-XX:+HeapDumpOnOutOfMemoryError',
'-Dcom.unboundid.directory.server.scriptName=setup'"
[21/Oct/2012:05:15:23.081 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555611 msg="Administrative alert type=server-starting
id=4178daee-ba3a-4be5-8e07-5ba17bf30b71
class=com.unboundid.directory.server.core.MetricsEngine
msg='The PingDataMetrics Server is starting'"
[21/Oct/2012:05:15:23.585 -0500] category=CORE severity=NOTICE
msgID=1879507338 msg="Starting group processing for backend api-users"
[21/Oct/2012:05:15:23.586 -0500] category=CORE severity=NOTICE
msgID=1879507339 msg="Completed group processing for backend api-users"
[21/Oct/2012:05:15:23.586 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555575 msg="'Group cache (2 static group(s) with 0 total
memberships and 0 unique members, 0 virtual static group(s),
1 dynamic group(s))' currently consumes 7968 bytes and can grow to a maximum
of an unknown number of bytes"
[21/Oct/2012:05:16:18.011 -0500] category=CORE severity=NOTICE
msgID=458887 msg="The PingDataMetrics Server (PingDataMetrics Server 7.0.1.0
build 20121021003738Z, R12799) has started successfully"
```

To Modify the File-Based Error Logs

- Use `dsconfig` to modify the default File-Based Error Log.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Error Logger" \
  --set include-product-name:true --set include-instance-name:true \
  --set include-startup-id:true
```

Managing the Syslog-Based Error Log Publisher

The Directory Server supports a Syslog-Based Error Log Publisher using the same mechanism as the Syslog-Based Access Log Publisher. You can easily configure the error logger using the `dsconfig` tool.

Syslog Error Mapping

The Directory Server automatically maps error log severities to the syslog severities. The following mappings are used:

Table 62: Error to Syslog Severities Mappings to Syslog

Error Log Facility	Syslog Severity
FATAL_ERROR,0	Syslog Emergency
SEVERE_ERROR,1	Syslog Alert
SEVERE_WARNING,2	Syslog Critical
MILD_ERROR,3	Syslog Error
MILD_WARNING,4	Syslog Warn
NOTICE,5	Syslog Notice
INFORMATION,6	Syslog Info
DEBUG,7	Syslog Debug

Configuring a Syslog-Based Error Log Publisher

You can configure a Syslog-based Error Log Publisher using the `dsconfig` tool. Again, we recommend that you use syslog locally on `localhost` and use `syslog-ng` to send the data packets over the UDP protocol.

Because syslog implementations differ by vendor, please review your particular vendor's syslog configuration.

To Configure a Syslog-Based Error Log Publisher

- Use `dsconfig` to create a log publisher of type `syslog-based-error`. In this example, set the syslog facility to 4 for security/authorization messages.

```
$ bin/dsconfig create-log-publisher --publisher-name "syslog-error" \
  --type syslog-based-error --set syslog-facility:4 --set enabled:true
```

Creating File-Based Debug Log Publishers

The Directory Server provides a File-Based Debug Log Publisher that can be configured when troubleshooting a problem that might occur during server processing. Because the debug data may be too large to maintain during normal operations, the Debug Log Publisher must be specifically configured and enabled. The Debug Log reports the following types of information:

Exception data thrown by the server.
Data read or written to network clients.
Data read or written to the database.
Access control or password policy data made within the server.

You can use the `dsconfig` tool to create a debug log publisher. You should only create a debug logger when troubleshooting a problem due to the voluminous output the Directory Server generates.

To Create a File-Based Debug Log Publisher

- Use `dsconfig` to create the debug log publisher. The `log-file` property (required) sets the debug log path. You must also specify the rotation and retention policy for the debug log.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "File-Based Debug Logger" \
  --type file-based-debug \
  --set enabled:true \
  --set log-file:/logs/debug \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy"
```

To Delete a File-Based Debug Log Publisher

- Use `dsconfig` to create the debug log publisher.

```
$ bin/dsconfig delete-log-publisher \
  --publisher-name "File-Based Debug Logger"
```

Chapter 21

Managing Monitoring

Topics:

The PingDirectory Server also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the PingDataMetrics Server, the Administrative Console, SNMP, JMX, and over LDAP. The Directory Server also provides a tool, the Periodic Stats Logger, to profile server performance.

This chapter presents the following information:

- *Profiling Server Performance Using the Stats Logger*
- *Adding Custom Logged Statistics to a Periodic Stats Logger*

The Monitor Backend

The Directory Server exposes its monitoring information under the cn=monitor entry. Administrators can use various means to monitor the servers, including the PingData Metrics Server, through SNMP, the Administrative Console, JConsole, LDAP command-line tools, and the Periodic Stats Logger. Use the bin/status tool to display server component activity and state.

The list of all monitor entries can be seen using ldapsearch as follows:

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
 --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
 --baseDN "cn=monitor" "(objectclass=*)" cn
```

The following table describes a subset of the monitor entries:

Table 63: Directory Server Monitoring Components

Component	Description
Active Operations	Provides information about the operations currently being processed by the Directory Server. Shows the number of operations, information on each operation, and the number of active persistent searches.
Backends	Provides general information about the state of an a Directory Server backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size.
Client Connections	Provides information about all client connections to the Directory Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., connID="15", connectTime="20100308223038Z", etc.)
Connection Handlers	Provides information about the available connection handlers on the Directory Server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
Disk Space Usage	Provides information about the disk space available to various components of the Directory Server.
General	Provides general information about the state of the Directory Server, including product name, vendor name, server version, etc.
Index	Provides on each index. The monitor captures the number of keys preloaded, and counters for read/write/remove/open-cursor/read-for-search. These counters provide insight into how useful an index is for a given workload.
HTTP/HTTPS Connection Handler Statistics	Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc.

Component	Description
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc.
System Information	Provides general information about the system and the JVM on which the Directory Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Directory Server version, including build ID, version, revision number, etc.
Work Queue	Provides information about the state of the Directory Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, and number of busy worker threads. The work queue configuration has a `monitor-queue-time` property set to `true` by default. This logs messages for new operations with a `qtime` attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

Monitoring Disk Space Usage

The disk space usage monitor provides information about the amount of usable disk space available for Directory Server components. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface by various components of the server. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `/config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the DiskSpaceConsumer interface.

The disk space usage monitor provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

Three thresholds can be configured for this monitor:

- **Low space warning threshold**. This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Directory Server will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The Directory Server will also generate additional alerts as the amount of usable space is further reduced (e.g., each time the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.
- **Low space error threshold**. This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. Once the server enters this mode, then an administrator will have to take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server

will resume normal operation. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.

- **Out of space error threshold**. This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the PingDirectory Server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.
- **Disk space monitoring for tools**. The server monitors disk space consumption during processing for the `export-ldif`, `rebuild-index`, and `backup` tools. Space is monitored every 10 seconds if usable space for all monitored paths is greater than 15 percent of the capacity of those volumes. If usable space for any path drops below 15 percent, or below 10GB free, the space check frequency is increased to every second. Warning messages are generated if available space falls below 10 percent, or below 5GB free. If usable space for any path drops below two percent, or 1GB free, the tool processing is aborted and files may be removed to free up space.

The default configuration uses the same values for the low space error threshold and out of space error threshold. This is to prevent having the server online but rejecting requests, which will cause problems with applications trying to interact with the server. The low space warning threshold generates an alert before the problem becomes serious, well in advance of available disk space dropping to a point that it is critical.

The default values may not be suitable for all disk sizes, and should be adjusted to fit the deployment. Determining the best values should factor in the size of the disk, how big the database may become, how much space log files may consume, and how many backups will be stored.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring with the PingDataMetrics Server

The PingDataMetrics Server is an invaluable tool for collecting, aggregating and exposing historical and instantaneous data from the various Ping Identity servers in a deployment. The PingDataMetrics Server relies on a captive PostgreSQL data store for the metrics, which it collects from internal instrumentation across the instances, replicas, and data centers in your environment. The data is available via a Monitoring API that can be used to build custom dashboards and monitoring applications to monitor the overall health of your PingData Platform system. For more information, see the *PingDataMetrics Server Administration Guide*.

About the Collection of System Monitoring Data

All PingDirectory Servers have the capability to monitor the health of the server and host system they run on for diagnostic review and troubleshooting. Initially, the servers do not collect any performance data until they are prepared for monitoring by a Metrics Server using the `monitored-servers add-servers` tool or an administrator enables system health data collection for real-time inspection and querying. At a high level, all of the important server and machine metrics which can be monitored are available in the `cn=monitor` backend.

The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics. Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and what kind of data aggregation occurs for LDAP application statistics. The Stats Collector plugin ensures that a PingDataMetrics Server is able to gather all of the detailed data required for a comprehensive diagnostic review.

The Stats Collector plugin relies exclusively on entries in the `cn=monitor` backend to sample data using LDAP queries. The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics. Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and the type of data aggregation that occurs for LDAP application statistics. The Stats Collector plugin is configured with the `dsconfig` tool and collects data using LDAP queries. For example, the `--server-info:extended` option includes collection for the following:

- CPU
- JVM memory
- Memory
- Disk information
- Network information

Utilization metrics are gathered via externally invoked OS commands, such as `iostat` and `netstat`, using platform-specific arguments and version-specific output parsing.

Enabling the Host System monitor provider automatically gathers CPU and memory utilization but only optionally gathers disk and network information. Disk and network interfaces are enumerated in the configuration by device names (e.g., `eth0` or lo), and by disk device names (e.g., `sd1, sdab, sda2, scsi0`).

Monitoring Key Performance Indicators by Application

The PingDirectory Server can be configured to track many key performance metrics (for example, throughput and response-time) by the client applications requesting them. This feature is invaluable for measuring whether the Ping Identity identify infrastructure meets all of your service-level agreements (SLA) that have been defined for client applications.

When enabled, the per-application monitoring data can be accessed in the `cn=monitor` backend, the Periodic Stats Logger, and made available for collection by the Metrics Server. See the “Profiling Server Performance Using the Periodic Stats Logger” for more information on using that component. Also, see the Directory Server Configuration section of the *PingData Metrics Server Administration Guide* for details on configuring the server to expose metrics that interest you. Tracked application information is exposed in the PingDataMetrics Server by metrics having the 'application-name' dimension. See the documentation under `docs/metrics` of the PingDataMetrics Server for information on which metrics are available with the 'application-name' dimension.

Configuring the External Servers

Before you install the PingDataMetrics Server, you need to configure the servers you will be monitoring: PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. The PingDataMetrics Server requires all servers to be version 3.5.0 or later. See the administration guides for each product for installation instructions.

Once you have installed the Directory Server, you can use the `dsconfig` tool to make configuration changes for the PingDataMetrics Server. When using the `dsconfig` tool interactively, set the complexity level to Advanced, so that you can make all the necessary configuration changes.

Preparing the Servers Monitored by the PingDataMetrics Server

The Metrics Backend manages the storage of metrics and provides access to the stored blocks of metrics via LDAP. The Metrics Backend is configured to keep a maximum amount of metric history based on log retention policies. The default retention policy uses the Default Size Limit Retention Policy, Free Disk Space Retention Policy, and the File Growth Limit Policy, limiting the total disk space used to 500 MB. This amount of disk typically contains more than

24 hours of metric history, which is ample. The Directory Server keeps a metric history so that the PingDataMetrics Server can be down for a period and then catch up when it comes back online.

The following two commands create a Retention Policy that limits the number of files to 2000, and sets the Metrics Backend to flush data to a new file every 30 seconds.

```
$ bin/dsconfig create-log-retention-policy \
  --policy-name StatsCollectorRetentionPolicy \
  --type file-count --set number-of-files:2000

$ bin/dsconfig set-backend-prop \
  --backend-name metrics --set sample-flush-interval:30s \
  --set retention-policy:StatsCollectorRetentionPolicy
```

These commands configure the Metrics Backend to keep 16 hours of metric history, which consumes about 250 MB of disk, ensuring that captured metrics are available to the PingDataMetrics Server within 30 seconds of when the metric was captured. The value of the `sample-flush-interval` attribute determines the maximum delay between when a metric is captured and when it can be picked up by the PingDataMetrics Server.

The flush interval can be set between 15 seconds and 60 seconds, with longer values resulting in less processing load on the PingDataMetrics Server. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in the Dashboard Application. If you change the `sample-flush-interval` attribute to 60 seconds in the example above, then the Directory Server keeps 2000 minutes of history. Because the number of metrics produced per unit of time can vary depending on the configuration, no exact formula can be used to compute how much storage is required for each hour of history. However, 20 MB per hour is a good estimate.

Configuring the Processing Time Histogram Plugin

The Processing Time Histogram plugin is configured on each Directory Server and Directory Proxy Server as a set of histogram bucket ranges. When the bucket ranges for a histogram change, the PingDataMetrics Server notices the change and marks samples differently. This process allows for histograms with the same set of bucket definitions to be properly aggregated and understood when returned in a query. If different servers have different bucket definitions, then a single metric query cannot return histogram data from the servers.

You should try to keep the Processing Time Histogram bucket definitions the same on all servers. Having different definitions restricts the ability of the PingDataMetrics Server API to aggregate histogram data across servers and makes the results of a query asking "What percentage of the search requests took less than 12 milliseconds?" harder to understand.

For each server in your topology, you must set the separate-monitor-entry-per-tracked-application property of the processing time histogram plugin to true. This property must be set to expose per-application monitoring information under `cn=monitor`. When the separate-monitor-entry-per-tracked-application property is set to true, then the per-application-ldap-stats property must be set to per-application-only on the Stats Collector Plugin and vice versa.

For example, the following `dsconfig` command line sets the required properties of the Processing Time Histogram plugin:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Processing Time Histogram” \
  --set separate-monitor-entry-per-tracked-application:true
```

The following `dsconfig` command line sets the per-application-ldap-stats property of the Stats Collector plugin to per-application-only:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Stats Collector” \
 --set per-application-ldap-stats:per-application-only
```

Setting the Connection Criteria to Collect SLA Statistics by Application

If you want to collect data about your SLAs, you need to configure connection criteria for each Service Level Agreement that you want to track. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used when the server needs to perform matching based on connection-level properties, such as filtered logging. For assistance using connection criteria, contact your authorized support provider.

For example, imagine that we are interested in collecting statistics on data that is accessed by clients authenticating as the Directory Manager. We need to create connection criteria on the Directory Server that identifies any user authenticating as the Directory Manager. The connection criteria name corresponds to the application-name dimension value that clients will specify when accessing the data via the API. When you define the Connection Criteria, change the included-user-base-dn property to include the Directory Manager’s full LDIF entry.

The following `dsconfig` command line creates connection criteria for the Directory Manager:

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name “Directory Manager” \
  --type simple \
  --set “included-user-base-dn:cn=Directory Manager,cn=Root DNs,cn=config”
```

Updating the Global Configuration

You also need to create Global Configuration-tracked applications for each app (connection criteria) you intend to track. The `tracked-application` property allows individual applications to be identified in the server by connection criteria. The name of the tracked application is the same as the name you defined for the connection criteria.

For example, the following `dsconfig` command line adds the connection criteria we created in the previous step to the list of tracked applications:

```
$ bin/dsconfig set-global-configuration-prop \
  --set "tracked-application:Directory Manager”
```

The value of the `tracked-application` field corresponds to the value of the `application-name` dimension value that clients will specify when accessing the data via the API.

Proxy Considerations for Tracked Applications

In a proxy environment, the criteria should be defined in the Directory Proxy Server since the Directory Proxy Server passes the application name through to the Directory Server in the intermediate client control. If a client of the Directory Proxy Server or Directory Server happens to use the intermediate client control, then the client name specified in the control will be used as the application name regardless of the criteria listed in the tracked-application property.

Monitoring Using SNMP

The PingDirectory Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems. Consult with your NMS system for specific information.

The PingDirectory Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Server’s SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

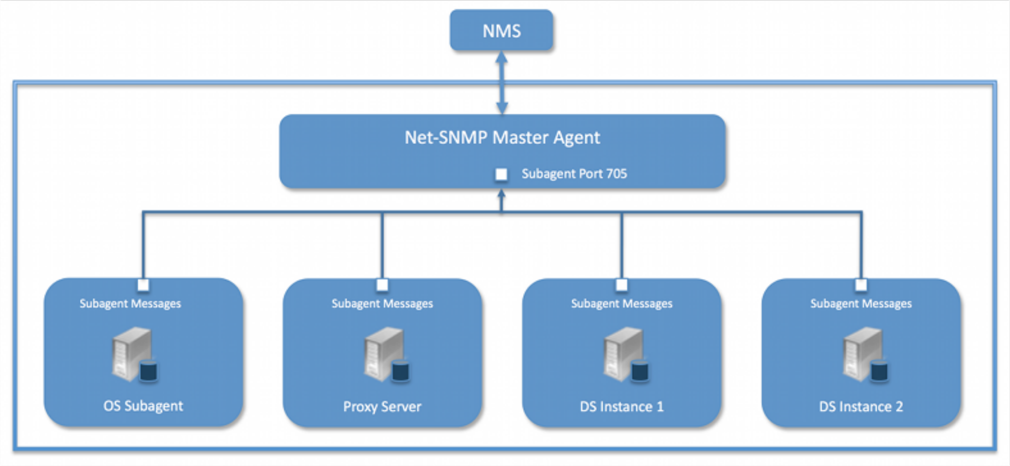


Figure 19: Example SNMP Deployment

One important note is that the PingDirectory Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note: The Directory Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

   ```
   $ bin/dsconfig set-alert-handler-prop \
     --handler-name "SNMP Subagent Alert Handler" --set enabled:true
   ```
2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

   ```
   The SNMP sub-agent was unable to connect to the master
   agent at localhost/705: Timeout
   ```
3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

   ```
   master agentx agentXSocket tcp:localhost:705
   ```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.
4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

   ```
   trap2sink localhost public 162
   ```
5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

   ```
   rwuser initial
   createUser initial MD5 setup_passphrase DES
   ```
6. Run the following command to create the SNMPv3 user.

   ```
   snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
   localhost create snmpuser initial
   ```
7. Start the `snmpd` daemon and after a few seconds you should see the following message in the Directory Server error log:

   ```
   The SNMP subagent connected successfully to the master agent
   at localhost:705. The SNMP context name is host.example.com:389
   ```
8. Set up a trap client to see the alerts that are generated by the Directory Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

   ```
   authcommunity log, execute public
   ```
9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

   ```
   $ cp resource/mib/* /usr/share/snmp/mibs
   ```
10. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (-f), do not read any configuration files (-C) except the one specified with the -c option, print to standard output (-Lo), and then specify that debugging output should be turned on for the User-based Security Module (-Dusm). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

    ```
    $ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
      -m all -M +/usr/share/snmp/mibs
    ```

11. Run the Net-SNMP client tools to test the feature. The following options are required: -v <SNMP version>, -u <user name>, -A <user password>, -l <security level>, -n <context name (instance name)> . The -m all option loads all MIBs in the default MIB directory in /usr/share/snmp/mibs so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389
 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n
 host.example.com:389 \
-m all localhost systemStatus
```

MIBS

The Directory Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Monitoring with the Administrative Console

Ping Identity has an Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.
4. Use the top level navigation dropdown and select 'Status.'
5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

The PingDirectory Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the Administrative Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The PingDirectory Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Server using the `dsconfig` tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

To Run JConsole

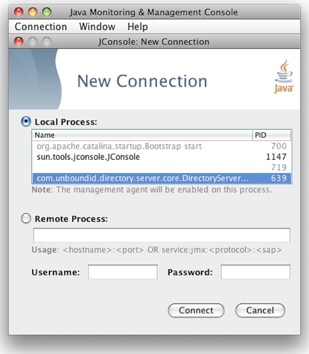
1. Use `JConsole` to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: `jconsole PID`. Or you can run JConsole remotely using: `jconsole hostname:port`.

   ```
   $ jconsole
   ```

   Note: If SSL is configured on the JMX Connection Handler, you must specify the Directory Server jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

   ```
   $ jconsole \
     -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
     -J-Djavax.net.ssl.trustStorePassword=secret \
     -J-Djava.class.path=$SERVER_ROOT/lib/PingDirectory.jar:/Library/Java/
   JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/jconsole.jar
   ```

2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.

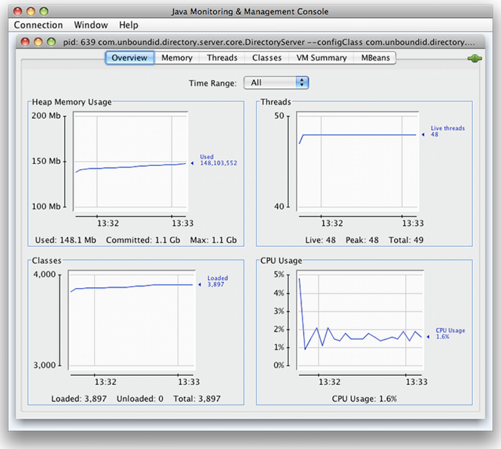


Name	PID
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1147
	719
com.unboundid.directory.server.core.DirectoryServer...	639

Note: The management agent will be enabled on this process.

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

3. Review the resource monitoring information.



pid: 639 com.unboundid.directory.server.core.DirectoryServer --configClass com.unboundid.directory....

Time Range: All

Used: 148.1 Mb Committed: 1.1 Gb Max: 1.1 Gb

Live: 48 Peak: 48 Total: 49

Loaded: 3,897 Unloaded: 0 Total: 3,897

CPU Usage: 1.6%

Monitoring the Directory Server Using JConsole

You can set up JConsole to monitor the Directory Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscibe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Server using JConsole

1. Start the Directory Server.

```
$ bin/start-server
```

2. Enable the JMX Connection handler using the `dsconfig` tool. The handler is disabled by default. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

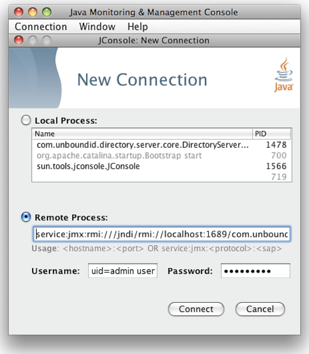
```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
```

```
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



○ Local Process:

Name	PID
com.unboundid.directory.server.core.DirectoryServer...	1478
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1566
	719

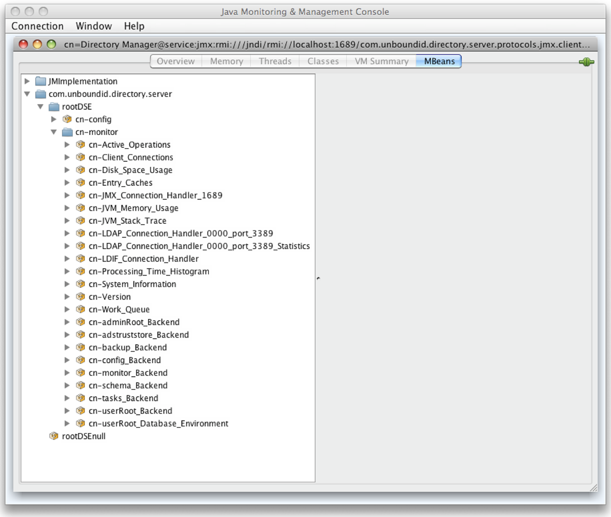
● Remote Process:

service:jmx:rmi:///jndi/rmi://localhost:1689/com.unbounc

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: uid=admin user Password: ••••••••••

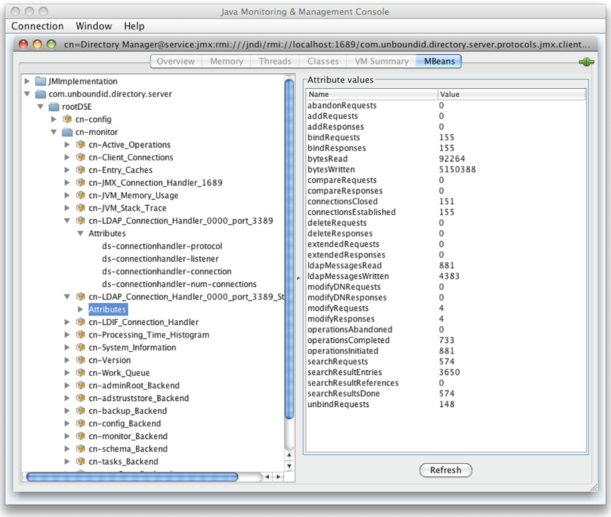
6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.



cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...

```
JMImplementation
com.unboundid.directory.server
  rootDSE
    cn-config
    cn-monitor
      cn-Active_Operations
      cn-Client_Connections
      cn-Disk_Space_Usage
      cn-Entry_Caches
      cn-JMX_Connection_Handler_1689
      cn-JVM_Memory_Usage
      cn-JVM_Stack_Trace
      cn-LDAP_Connection_Handler_0000_port_3389
      cn-LDAP_Connection_Handler_0000_port_3389_Statistics
      cn-LDIF_Connection_Handler
      cn-Processing_Time_Histogram
      cn-System_Information
      cn-Version
      cn-Work_Queue
      cn-adminRoot_Backend
      cn-adstruststore_Backend
      cn-backup_Backend
      cn-config_Backend
      cn-monitor_Backend
      cn-schema_Backend
      cn-tasks_Backend
      cn-userRoot_Backend
      cn-userRoot_Database_Environment
  rootDSEnull
```

7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.


Java Monitoring & Management Console
Connection
Window
Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview
Memory
Threads
Classes
VM Summary
MBeans
Refresh

- JMImplementation
- com.unboundid.directory.server
  - rootDSE
    - cn-config
    - cn-monitor
      - cn-Active_Operations
      - cn-Client_Connections
      - cn-Entry_Caches
      - cn-JMX_Connection_Handler_1689
      - cn-JVM_Memory_Usage
      - cn-JVM_Stack_Trace
      - cn-LDAP_Connection_Handler_0000_port_3389
        - Attributes
          - ds-connectionhandler-protocol
          - ds-connectionhandler-listener
          - ds-connectionhandler-connection
          - ds-connectionhandler-num-connections
      - cn-LDAP_Connection_Handler_0000_port_3389_St
        - Attributes
      - cn-LDIF_Connection_Handler
      - cn-Processing_Time_Histogram
      - cn-System_Information
      - cn-Version
      - cn-Work_Queue
      - cn-adminRoot_Backend
      - cn-adstruststore_Backend
      - cn-backup_Backend
      - cn-config_Backend
      - cn-monitor_Backend
      - cn-schema_Backend
      - cn-tasks_Backend

Attribute values

Name	Value
abandonRequests	0
addRequests	0
addResponses	0
bindRequests	155
bindResponses	155
bytesRead	92264
bytesWritten	5150388
compareRequests	0
compareResponses	0
connectionsClosed	151
connectionsEstablished	155
deleteRequests	0
deleteResponses	0
extendedRequests	0
extendedResponses	0
ldapMessagesRead	881
ldapMessagesWritten	4383
modifyDNRequests	0
modifyDNResponses	0
modifyRequests	4
modifyResponses	4
operationsAbandoned	0
operationsCompleted	733
operationsInitiated	881
searchRequests	574
searchResultEntries	3650
searchResultReferences	0
searchResultsDone	574
unbindRequests	148

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " +
 String.valueOf(value));
      }
     }
     System.out.println();
  }
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring over LDAP

The PingDirectory Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Profiling Server Performance Using the Stats Logger

The Directory Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Directory Server ships with the built-in "Stats Logger' disabled. To enable it using the `dsconfig` tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select .

1. Run `dsconfig` in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

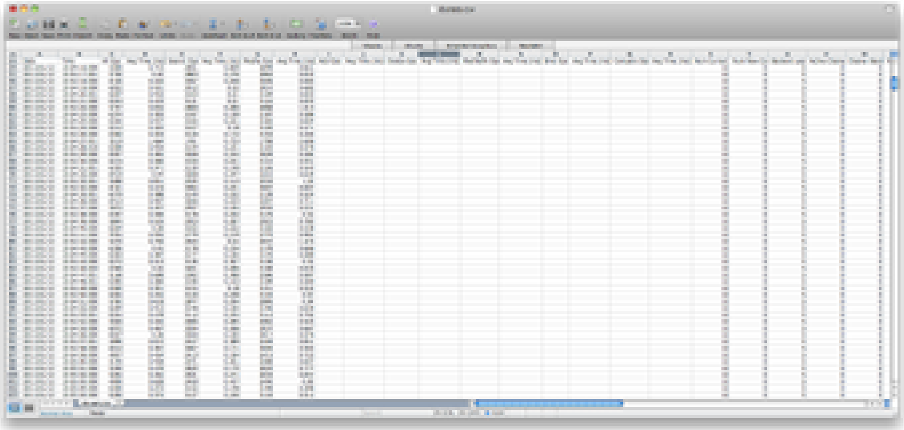
```
$ bin/dsconfig
```

2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plug-in.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

```
>>>> Configure the properties of the Stats Logger Plugin

Property                Value(s)
------------------------------------------------------------------------
1)   description        Logs performance stats to a log file
                        periodically.
2)   enabled                     false
3)   local-db-backend-info       basic
4)   replication-info            basic
5)   entry-cache-info            -
6)   host-info                   -
7)   included-ldap-application   If per-application LDAP stats is enabled,
                                 then stats will be included for all
                                 applications.
8)   log-interval                1 s
9)   collection-interval         200 ms
10)  suppress-if-idle            true
11)  header-prefix-per-column    false
12)  empty-instead-of-zero       true
13)  lines-between-header        50
14)  included-ldap-stat          active-operations, num-connections,
                                 op-count-and-latency, work-queue
15)  included-resource-stat      memory-utilization
16)  histogram-format            count
17)  histogram-op-type           all
18)  per-application-ldap-stats  aggregate-only
19)  ldap-changelog-info         -
20)  gauge-info                  none
21)  log-file                    logs/dsstats.csv
```

```
22)   log-file-permissions          640
23)   append                        true
24)   rotation-policy               Fixed Time Rotation Policy, Size Limit
                                    Rotation Policy
25)   retention-policy              File Count Retention Policy

?)    help
f)    finish - apply any changes to the Periodic Stats Logger Plugin
a)    hide advanced properties of the Periodic Stats Logger Plugin
d)    display the equivalent dsconfig command lines to either re-create this
            object or only to apply pending changes
b)    back
q)    quit

Enter choice [b]:
```

7. Run the Directory Server. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate` --help to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in *To Enable the Stats Logger*.
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. On the Directory Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.

12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                   Value(s)
         --------------------------------------------------------------
    1)   description                -
    2)   enabled                    true
    3)   monitor-objectclass        ds-memory-usage-monitor-entry
    4)   include-filter             -
    5)   attribute-to-log           total-bytes-used-by-memory-consumers
    6)   column-name                Memory Consumer Total (GB)
    7)   statistic-type             raw
    8)   header-prefix              -
    9)   header-prefix-attribute    -
    10)  regex-pattern              -
    11)  regex-replacement          -
    12)  divide-value-by            1073741824
    13)  divide-value-by-attribute  -
    14)  decimal-format             #.##
    15)  non-zero-implies-not-idle  false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

Chapter 22

Managing Notifications and Alerts

Topics:

The PingDirectory Server provides delivery mechanisms for account status notifications and administrative alerts using SMTP, JMX, or SNMP in addition to standard error logging. Alerts and events reflect state changes within the server that may be of interest to a user or monitoring service. Notifications are typically the delivery of an alert or event to a user or monitoring service. Account status notifications are only delivered to the account owner notifying a change in state in the account.

This chapter presents the following topics:

Working with Account Status Notifications

The PingDirectory Server supports notification handlers that can be used to notify users and/or administrators of significant changes related to password policy state for user entries. The following two notification handlers are available:

- **Error Log Account Status Notification Handler**. Enabled by default. The handlers send alerts to the error log when an account event occurs.
- **SMTP Account Status Notification Handler**. Disabled by default. You can enable the SMTP Handler with the `dsconfig` command to send notifications to designated email addresses.

Account Status Notification Types

The handlers send alerts when one of the account status events described in the following table occurs during password policy processing.

Table 64: Account Status Notification Types

Account Status Notification Types	Description
account-disabled	Generates a notification whenever a user account is disabled by an administrator.
account-enabled	Generates a notification whenever a user account is enabled by an administrator.
account-expired	Generates a notification whenever a user authentication attempt fails because the account has expired.
account-idle-locked	Generates a notification whenever a user authentication attempt fails because the account has been locked after idling for too long.
account-permanently-locked	Generates a notification whenever a user account is permanently locked (requiring administrative action to unlock the account) after too many failed attempts.
account-reset-locked	Generates a notification whenever an authentication attempt fails because the user account is locked because the user failed to change a password within the required interval that was reset by an administrator.
account-temporarily-locked	Generates a notification whenever a user account is temporarily locked after too many failed attempts.
account-unlocked	Generates a notification whenever a user account is unlocked by an administrator.
password-changed	Generates a notification whenever a user changes his or her own password.
password-expired	Generates a notification whenever a user authentication fails because the password has expired.
password-expiring	Generates a notification the first time that a password expiration warning is encountered for a user password.
password-reset	Generates a notification whenever a user's password is reset by an administrator.

Working with the Error Log Account Status Notification Handler

The Error Log Account Status Notification Handler is enabled by default and sends alerts when one of the account status events occur.

To Disable the Error Log Account Status Notification Handler

- Use the `dsconfig` tool to disable the Error Log Handler. You can view the log at `logs/error`.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  ---handler-name "Error Log Handler" --set enabled:false
```

To Remove a Notification Type from the Error Log Handler

- While not recommended, if you want to remove an account status notification type, use the `dsconfig` tool with the --remove option.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  --handler-name "Error Log Handler" \
  --remove account-status-notification-type: password-reset
```

Working with the SMTP Account Status Notification Handler

You can enable account status notifications to be sent to designated email addresses of end users, administrators, or both through an outgoing SMTP server. The email message is automatically generated from template files that contain the text to use in the message body. For example, the message subject for the account-disabled event is:

```
account-disabled: Your directory account has been disabled.
```

The message templates are located in the `config/messages` directory. The typical message body template is as follows:

```
Your directory account has been disabled.

For further assistance, please contact a server administrator.
```

By default, the sender address is notifications@example.com, but you can configure your own address.

Before you enable the SMTP Account Status Notification Handler, you must configure the Directory Server to use at least one mail server as shown below. You can configure an SMTP server using `dsconfig` and the set-global-configuration-prop option.

To Configure the SMTP Server

1. Use `dsconfig` to configure a simple mail server.

```
$ bin/dsconfig create-external-server --server-name smtp1 \
  --type smtp --set server-host-name:smtp.example.com
```

2. Use `dsconfig` to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  --set smtp-server:smtp1
```

To Configure a StartTLS Connection to the SMTP Server

1. Use `dsconfig` to configure a StartTLS connection to the server.

```
$ bin/dsconfig create-external-server \
  --server-name myTLSServer --type smtp \
  --set server-host-name:tls.smtp.example.com \
  --set server-port:587 \
  --set smtp-security:starttls \
  --set user-name:MyAccountName \
  --set password:AAD5yZ+DjvwiYkBSMer6GQ6B3szQ6gSSBjA=
```

2. Use `dsconfig` to configure a newly-created SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  -set smtp-server:myTLSServer
```

To Configure an SSL Connection to the SMTP Server

1. Use `dsconfig` to create an external SMTP server using SSL.

```
$ bin/dsconfig create-external-server \
  --server-name ssl.smtp.example.com \
  --type smtp --set server-host-name:stmp.gmail.com \
  --set server-port:465 \
  --set smtp-security:ssl \
  --set ’username:my.name@example.com’ \
  --set password:xxxxxx --set "smtp-timeout:10s"
```

2. Use `dsconfig` to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  --set smtp-server:ssl.smtp.example.com
```

To Enable the SMTP Account Status Notification Handler

- Use `dsconfig` to enable the SMTP account status notification handler.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  --handler-name "SMTP Handler" --set enabled:true \
  --set "recipient-address:admin@example.com" \
  --set "sender-address:acct-status-notifications@example.com
```

To View the Account Status Notification Handlers

- After you have enabled the SMTP server, view the list of account status notification handlers using `dsconfig`.

```
$ bin/dsconfig list-account-status-notification-handlers
```

```
Account Status Notification Handler : Type   : enabled
------------------------------------:-----------:--------
Error Log Handler                : error-log : true
SMTP Handler                     : smtp   : true
```

Associating Account Status Notification Handlers with Password Policies

To generate notifications whenever appropriate password policy state changes occur in the server, the password policy that governs the entry being updated must be configured to use one or more account status notification handlers. By default, password policies are not configured with any such handlers, and therefore, no account status notifications will be generated.

The set of account status notification handlers that should be in use for a password policy is controlled by the `account-status-notification-handler` property for that password policy. It can be configured using `dsconfig` or the Administrative Console. For example, the following change updates the default password policy, so that the error log account status notification handler will be invoked for any appropriate password policy state changes for entries governed by the default password policy:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "account-status-notification-handler:Error Log Handler"
```

Working with Administrative Alert Handlers

The PingDirectory Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler**. Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Directory Server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler**. Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. Ping Identity uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler**. Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler**. Provides alert handler implementations created in third-party code using the Server SDK.

Administrative Alert Types

If enabled, the Directory Server can generate administrative alerts when the events occur. A full listing of system alerts and their severity is available in `<server-root>/docs/admin-alerts-list.csv`

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the `dsconfig` tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using `dsconfig` in interactive command-line mode, which is visible on the "Standard" object menu.

To Configure the JMX Connection Handler

1. Use `dsconfig` to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the `ldapmodify` tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
```

```
givenName: JMX
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

To Configure the JMX Alert Handler

- Use `dsconfig` to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the `dsconfig` tool.

Configuring the SMTP Alert Handler

- Use the `dsconfig` tool to configure the SMTP Alert Handler.

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the `dsconfig` tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP2c. For more information on SNMP, see Monitoring Using SNMP.

To Configure the SNMP Subagent Alert Handler

- Use `dsconfig` to configure the SNMP subagent alert handler. The `server-host-name` is the address of the system running the SNMP subagent. The `server-port` is the port number on which the subagent is running. The `community-name` is the name of the SNMP community that is used for the traps.

  The Directory Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

Working with the Alerts Backend

The Directory Server stores recently generated admin alerts in an Alerts Backend under the `cn=alerts` branch. The backend makes it possible to obtain admin alert information over LDAP for use with remote monitoring. The backend's primary job is to process search operations for alerts. It does not support add, modify, or modify DN operations of entries in the `cn=alerts` backend.

The alerts persist on disk in the `config/alerts.ldif` file so that they can survive server restarts. By default, the alerts remain on disk for seven days before being removed. However, administrators can configure the number of days for alert retention using the `dsconfig` tool. The administrative alerts of Warning level or worse that have occurred in the last 48 hours are viewable from the output of the status command-line tool and in the Administrative Console.

To View Information in the Alerts Backend

- The following uses `ldapsearch` to view the admin alerts.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alerts "(objectclass=ds-admin-alert)"
```

```
dn: ds-alert-id=3d1857a2-e8cf-4e80-ac0e-ba933be59eca,cn=alerts
objectClass: top
objectClass: ds-admin-alert
ds-alert-id: 3d1857a2-e8cf-4e80-ac0e-ba933be59eca
ds-alert-type: server-started
ds-alert-severity: info
ds-alert-type-oid: 1.3.6.1.4.1.32473.2.11.33
ds-alert-time: 20110126041442.622Z
ds-alert-generator: com.unboundid.directory.server.core.directory.server
ds-alert-message: The Directory Server has started successfully
```

To Modify the Alert Retention Time

1. Use `dsconfig` to change the maximum time information about generated admin alerts is retained in the Alerts backend. After this time, the information gets purged from the Directory Server. The minimum retention time is 0 milliseconds, which immediately purges the alert information.

```
$ bin/dsconfig set-backend-prop --backend-name "alerts" \
  --set "alert-retention-time: 2 weeks"
```

2. View the property using `dsconfig`.

```
$ bin/dsconfig get-backend-prop --backend-name "alerts" \
  --property alert-retention-time
```

```
Property : Value(s)
---------------------:---------
alert-retention-time : 2 w
```

To Configure Duplicate Alert Suppression

- Use `dsconfig` to configure the maximum number of times an alert is generated within a particular timeframe for the same condition. The `duplicate-alert-time-limit` property specifies the length of time that must pass before duplicate messages are sent over the administrative alert framework. The `duplicate-alert-limit` property specifies the maximum number of duplicate alert messages should be sent over the administrative alert framework in the time limit specified in the `duplicate-alert-time-limit` property.

```
$ bin/dsconfig set-global-configuration-prop \
  --set duplicate-alert-limit:2 \
  --set "duplicate-alert-time-limit:3 minutes"
```

Working with Alarms, Alerts, and Gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server unavailable' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and Administrative Console. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The Directory Server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

The Directory Server is compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when the Condition and Resource properties are the same. The Condition corresponds to the Summary column in the `admin-alerts-list.csv` file.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms.

To View Information in the Alarms Backend

- The following uses `ldapsearch` to view alarms. The following displays the listing for the CPU usage alarm.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alarms "(objectclass=ds-admin-alarm)"
```

```
dn: ds-alarm-id=CPU Usage (Percent)-Host System,cn=alarms
dn: ds-alarm-id=CPU Usage (Percent)-Host System,cn=alarms
objectClass: top
objectClass: ds-admin-alarm
ds-alarm-id: CPU Usage (Percent)-Host System
ds-alarm-condition: CPU Usage (Percent)
ds-alarm-specific-resource: Host System
ds-alarm-severity: CRITICAL
ds-alarm-previous-severity: CRITICAL
ds-alarm-details: Gauge CPU Usage (Percent) for Host System
       has value 99, having had a value of 83.13 in the
       previous interval. The severity is critical, having
       assumed this severity Thu Sep 25 10:24:20 CDT 2014
       when the value crossed threshold 80
```

```
ds-alarm-additional-text: If CPU use is high, check the server's current
 workload
      and other processes on this system and make any needed adjustments.
 Reducing
      the load on the system will lead to better response times
ds-alarm-start-time: 20140925152420.004Z
ds-alarm-critical-last-time: 20140925152420.004Z
ds-alarm-critical-total-duration-millis: 0
```

Testing Alerts and Alarms

After alarms and alert handlers are configured, verify that the server takes the appropriate action when an alarm state changes by manually increasing the severity of a gauge. Alarms and alerts can be verified with the `status` tool.

To Test Alarms and Alerts

1. Configure a gauge with `dsconfig` and set the `override-severity` property to critical. The following example uses the CPU Usage (Percent) gauge.

```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

2. Run the `status` tool to verify that an alarm was generated with corresponding alerts. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status

                        --- Administrative Alerts ---
 Severity : Time            : Message
 ---------:-----------------:------------------------------------------------------
 Info     : 11/Aug/2014     : A configuration change has been made in the
          : 15:48:46 -0500  : Directory Server:
          :                 : [11/Aug/2014:15:48:46.054 -0500]
          :                 : conn=17 op=73 dn='cn=Directory Manager,cn=Root
          :                 : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                 : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                 :  --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                 : --set warning-value:-1'
 Info     : 11/Aug/2014     : A configuration change has been made in the
          :  15:47:32 -0500 : Directory Server: [11/Aug/2014:15:47:32.547
 -0500]
          :                 : conn=4 op=196 dn='cn=Directory Manager,cn=Root
          :                 : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                 : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                 : --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                 :  --set warning-value:0'
 Error    : 11/Aug/2014     : Alarm [CPU Usage (Percent). Gauge CPU Usage
 (Percent)
          :  15:41:00 -0500 : for Host System has
          :                 : a current value of '18.583333333333332'.
          :                 : The severity is currently OVERRIDDEN in the
          :                 : Gauge's configuration to 'CRITICAL'.
          :                 : The actual severity is: The severity is
          :                 : currently 'NORMAL', having assumed this
 severity
```

```
          :                 : Mon Aug 11 15:41:00 CDT 2014. If CPU use is
 high,
          :                 : check the server's current workload and make
 any
          :                 : needed adjustments. Reducing the load on the
 system
          :                 : will lead to better response times.
          :                 : Resource='Host System']
          :                 : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
 hours
 Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
                         --- Alarms ---
 Severity : Severity Start : Condition : Resource    : Details
          : Time           :           :             :
 ---------:----------------:-----------:-------------:------------------------------
 Critical : 11/Aug/2014    : CPU Usage : Host System : Gauge CPU Usage
 (Percent) for
          : 15:41:00 -0500 : (Percent) :             : Host System
          :                :           :             : has a current value
 of
          :                :           :             : '18.785714285714285'.
          :                :           :             : The severity is
 currently
          :                :           :             : 'CRITICAL', having
 assumed
          :                :           :             : this severity Mon Aug
 11
          :                :           :             : 15:49:00 CDT 2014. If
 CPU use
          :                :           :             : is high, check the
 server's
          :                :           :             : current workload and
 make any
          :                :           :             : needed adjustments.
 Reducing
          :                :           :             : the load on the
 system will
          :                :           :             : lead to better
 response times
 Warning  : 11/Aug/2014    : Work Queue: Work Queue  : Gauge Work Queue Size
 (Number
          : 15:39:40 -0500 : Size      :             : of Requests) for Work
 Queue
          :                : (Number of:             : has a current value
 of '27'.
          :                : Requests) :             : The severity is
 currently
          :                :           :             : 'WARNING' having
 assumed this
          :                :           :             : severity Mon Aug 11
 15:48:50
          :                :           :             : CDT 2014. If all
 worker
          :                :           :             : threads are busy
 processing
          :                :           :             : other client
 requests, then
          :                :           :             : new requests that
 arrive will
          :                :           :             : be forced to wait in
 the work
```

```
          :                :           :             : queue until a worker
 thread
          :                :           :             : becomes available
Shown are alarms of severity [Warning,Minor,Major,Critical]
Use the --alarmSeverity option to filter this list
```

Indeterminate Alarms

Indeterminate alarms are raised for a server condition for which a severity cannot be determined. In most cases these alarms are benign and do not issue alerts nor appear in the output of the `status` tool or Administrative Console by default. These alarms are usually caused by an enabled gauge that is intended to measure an aspect of the server that is not currently enabled. For example, gauges intended to monitor metrics related to replication may produce indeterminate alarms if a Directory Server is not currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There may be messages that can help determine the issue. The following is sample output from the `status` tool run with the –`alarmSeverity=indeterminate` option:

```
                        --- Alarms ---
Severity     : Severity Start : Condition      : Resource   : Details
             : Time           :                :            :
-------------:----------------:----------------:------------:------------------------
Normal       : 26/Aug/2014    : Startup Begun  : cn=config  : The Directory
 Server
             : 14:16:29 -0500 :                :            : is starting.
             :                :                :            :
Indeterminate: 26/Aug/2014    : Replication    : not        : The value of
 gauge
             : 14:16:40 -0500 : Latency        : available  : Replication
 Latency
             :                : (Milliseconds) :            : (Milliseconds)
 could not
             :                :                :            : be determined.
 The
             :                :                :            : severity is
 INDETERMINATE,
             :                :                :            : having assumed
 this
             :                :                :            : severity Tue Aug
 26
             :                :                :            : 14:17:10 CDT
 2014.
```

The following is an indeterminate alarm for the Replication Latency (Milliseconds) gauge. The following is a sample search of the monitor backend for this gauge's entry. The result is an error message may explain the indeterminate severity:

```
# ldapsearch -w password --baseDN "cn=monitor"  \
-D"cn=directory manager" gauge-name="Replication Latency (Milliseconds)"

dn: cn=Gauge Replication Latency (Milliseconds),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-numeric-gauge-monitor-entry
objectClass: ds-gauge-monitor-entry
objectClass: extensibleObject
cn:          Gauge Replication Latency (Milliseconds)
gauge-name:  Replication Latency (Milliseconds)
resource:
severity:    indeterminate
summary:     The value of gauge Replication Latency (Milliseconds) could not
             be determined. The severity is INDETERMINATE, having assumed
             this severity Tue Aug 26 15:42:40 CDT 2014
error-message: No entries were found under cn=monitor having object
               class ds-replica-monitor-entry
              …
```

Chapter

23

Managing the SCIM Servlet Extension

Topics:

The PingDirectory Server provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS.

This section describes fundamental SCIM concepts and provides information on configuring SCIM on your server.

Overview of SCIM Fundamentals

Understanding the basic concepts of SCIM can help you use the SCIM extension to meet the your deployment needs. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension implements the 1.1 version of the SCIM specification. Familiarize yourself with this specification to help you understand and make efficient use of the SCIM extension and the SCIM SDK. The SCIM specifications are located on the Simplecloud website.

Summary of SCIM Protocol Support

PingDirectory Server implements all required features of the SCIM protocol and most optional features. The following table describes SCIM features and whether they are supported.

Table 65: SCIM Protocol Support

SCIM Feature	Supported
Etags	Yes
JSON	Yes
XML*	Yes
Authentication/Authorization	Yes, via HTTP basic authentication or OAuth 2.0 bearer tokens
Service Provider Configuration	Yes
Schema	Yes
User resources	Yes
Group resources	Yes
User-defined resources	Yes
Resource retrieval via GET	Yes
List/query resources	Yes
Query filtering*	Yes
Query result sorting*	Yes
Query result pagination*	Yes (Directory Server, not Directory Proxy Server)
Resource updates via PUT	Yes
Partial resource updates via PATCH*	Yes
Resource deletes via DELETE	Yes
Resource versioning*	Yes (requires configuration for updated servers)
Bulk*	Yes
HTTP method overloading	Yes
Raw LDAP Endpoints**	Yes

* denotes an optional feature of the SCIM protocol.

** denotes a PingDirectory Server extension to the basic SCIM functionality.

About the Identity Access API

The PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server support an extension to the SCIM standard called the Identity Access API. The Identity Access API provides an alternative to LDAP by supporting CRUD (create, read, update, and delete) operations to access directory server data over an HTTP connection.

SCIM and the Identity Access API are provided as a unified service through the SCIM HTTP Servlet Extension. The SCIM HTTP Servlet Extension can be configured to only enable core SCIM resources (e.g., 'Users' and 'Groups'), only LDAP object classes (e.g., `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there may be two representations with different schemas for any resources defined in the `scim-resources.xml` file: the SCIM representation and the raw LDAP representation. Likewise, because resources are exposed by an LDAP object class, and because these are hierarchical (e.g., `top` --> `person` --> `organizationalPerson` --> `inetOrgPerson`, etc.), a client application can access an entry in multiple ways due to the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the PingDirectory Server.

Configuring SCIM

This section discusses details about the PingDirectory Server implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available on the Simplecloud website.

Creating Your Own SCIM Application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM SDK is available for download at https://github.com/pingidentity/scim.

Note: The value of a read-only SCIM attribute can be set by a POST operation if the SCIM attribute is a custom attribute in the `scim-resource.xml` config file, but not if the SCIM attribute is a core SCIM attribute.

Configuring the SCIM Servlet Extension

The Directory Server provides a default SCIM HTTP Servlet Extension that can be enabled and configured using a `dsconfig` batch script located in the `config` directory. The script runs a series of commands that enables an HTTPS Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension. There are additional optional configurations (e.g., changing the log format, enable `entryDN` virtual attribute and using VLV indexes) that you can make by altering the `dsconfig` batch script.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See *Managing the SCIM Schema* for more information.

The following procedures show how to configure SCIM on the server. The first example procedure shows the steps to manually configure SCIM without using the script. The second example procedure uses the `dsconfig` batch script to configure SCIM.

To Configure SCIM Manually

The following example procedure assumes that you have configured the Directory Server using the default settings, which means that SSL and the HTTPS Connection Handler have not been configured. The example also shows the `dsconfig` non-interactive commands. You can easily use the `dsconfig` interactive commands, which uses a menu-driven interface. If you use the `dsconfig` interactive, you must change to the Standard or Advanced object menu to access many of these configuration settings.

1. Set up your certificates. Follow the examples shown in the section *Managing Certificates*. You should have a keystore and truststore set up in the `config` directory. Make sure that the keystore.pin and truststore.pin are set.
2. Enable the key manager provider. The key manager provider accesses the certificate during the SSL handshaking process. If running `dsconfig` interactive, open the main menu, select "Key Manager Provider" -> "View and edit an existing Key Manager Provider" -> "JKS" (or the type of certificate you are working with) -> "enabled" and then set the value to "true". Click "finish" to save the setting.

```
$ bin/dsconfig create-key-manager-provider --provider-name JKS \
  --type file-based --set enabled:true --set key-store-file:config/keystore
 \
  --set key-store-type:JKS --set key-store-pin-file:config/keystore.pin
```

3. Enable the trust manager provider. The trust manager provider determines if a presented certificate can be trusted. If running `dsconfig` interactive, open the main menu, select "Trust Manager Provider" -> "View and edit an existing Trust Manager Provider" -> "JKS" (or the type of certificate you are working with) -> "enabled" and then set the value to "true". Click "finish" to save the setting.

```
$ bin/dsconfig create-trust-manager-provider --provider-name JKS \
  --type file-based --set enabled:true --set trust-store-file:config/
truststore \
  --set trust-store-type:JKS --set trust-store-pin-file:config/
truststore.pin
```

4. Configure the HTTPS Connection Handler. The command enables the connection handler, specifies the SCIM HTTP servlet extension and sets the listen port to a port of your choice, in this example, use 8443. The command also specifies the type of key manager and trust manager providers and sets the log publisher to "HTTP Detailed Access." If running `dsconfig` interactive, open the main menu, select "Connection Handler" -> "View and edit an existing Connection Handler" -> "HTTPS Connection Handler". Change the parameters to match your setup, and then, click "finish" to save the setting.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "HTTPS Connection Handler" \
  --type http --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set http-servlet-extension:SCIM \
  --set "http-operation-log-publisher:HTTP Detailed Access" \
  --set key-manager-provider:JKS --set trust-manager-provider:JKS
```

5. Add access controls to allow access to LDAP controls used by the SCIM Servlet Extension. These controls are the Post-Read Request Control (1.3.6.1.1.13.2), Server-Side Sort Request Control (1.2.840.113556.1.4.473), Simple Paged Results Control (1.2.840.113556.1.4.319), and Virtual List View Request Control (2.16.840.1.113730.3.4.9). We recommend using the following command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473
 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9")
    (version 3.0;acl "Authenticated access to controls used by the SCIM
 servlet
     extension"; allow (all) userdn="ldap:///all";)'
```

6. Add access controls to allow read access to operational attributes used by the SCIM Servlet Extension. We recommend using the following non-interactive command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id
 ||
    createTimestamp || modifyTimestamp")
    (version 3.0;acl "Authenticated read access to operational attributes \
    used by the SCIM servlet extension"; allow (read,search,compare)
    userdn="ldap:///all";)'
```

7. Optional. The SCIM HTTP Connection Handler automatically uses a detailed HTTP log publisher, which is implemented in a proprietary format. If you need a standard W3C common log format publisher, enter the following command. If running `dsconfig` interactive, open the main menu, select "Log Publisher" -> "Create a new Log Publisher" -> "new Log Publisher created from scratch" -> "File Based Access Log Publisher", enter the parameters to match your setup, and then, click "finish" to save the setting. Go back to the main menu, select "Connection Handler" -> "HTTPS Connection Handler", and then add "HTTP Common Access" to the `http-operation-log-publisher` property. Click "finish" to save the setting.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Common Access" \
  --type common-log-file-http-operation --set enabled:true \
  --set log-file:logs/http-common-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"

$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add "http-operation-log-publisher:HTTP Common Access"
```

8. Optional. To support searching or filtering by DN using the Identity Access API, you can enable the entryDN virtual attribute. If running `dsconfig` interactive, open the main menu, select "Virtual Attribute" -> "View and edit an existing Virtual Attribute" -> "Entry DN", and then change the enabled property to "true". Click "finish" to save the setting.

```
$ bin/dsconfig set-virtual-attribute-prop --name entryDN --set enabled:true
```

9. Optional. To support pagination, create some Virtual List View indexes. If running `dsconfig` interactive, open the main menu, select "Local DB VLV Index" -> "Create a new Local DB VLV Index" and then enter the properties needed for your setup. Click "finish" to save the setting. Repeat again for the "ascending-sn" index. Then, run the `rebuild-index` command to let the VLV Indexes take effect.

```
$ bin/dsconfig create-local-db-vlv-index --backend-name userRoot \
--index-name ascending-uid --set base-dn:dc=example,dc=com \
--set scope:whole-subtree --set "filter:(objectclass=inetorgperson)" \
--set "sort-order:+uid"

$ bin/dsconfig create-local-db-vlv-index --backend-name userRoot \
--index-name ascending-sn --set base-dn:dc=example,dc=com \
--set scope:whole-subtree --set "filter:(objectclass=inetorgperson)" \
--set "sort-order:+sn"

$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.ascending-uid \
--index vlv.ascending-sn
```

To Enable Resource Versioning

Resource versioning is enabled by default in new installations. Upgraded servers that had SCIM enabled need additional configuration to enable resource versioning.

1. Enable the `ds-entry-checksum` virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
    --name ds-entry-checksum \
    --set enabled:true
```

2. Remove any existing access controls required by SCIM for read access to operational attributes:

```
$ bin/dsconfig set-access-control-handler-prop \
    --remove 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-
unique-id || createTimestamp || ds-create-time || modifyTimestamp || ds-
update-time")(version 3.0;acl "Authenticated read access to operational
 attributes used by the SCIM servlet extension"; allow (read,search,compare)
 userdn="ldap:///all"'
```

3. Add new access controls required by SCIM for read access to operational attributes with the addition of the `ds-entry-checksum`:

```
$ bin/dsconfig set-access-control-handler-prop \
    --add  'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-
id || createTimestamp || ds-create-time || modifyTimestamp || ds-update-
time || ds-entry-checksum")(version 3.0;acl "Authenticated read access
 to operational attributes used by the SCIM servlet extension"; allow
 (read,search,compare) userdn="ldap:///all"'
```

4. Enable SCIM resource versioning using the entry checksum virtual attribute:

```
$ bin/dsconfig set-http-servlet-extension-prop \
    --extension-name SCIM \
    --set entity-tag-ldap-attribute:ds-entry-checksum
```

   If enabled, the value of the `ds-entry-checksum` attribute is returned as the `ETag` header value when accessing the resource through SCIM, and is checked against the `If-Match` header when updating the resource. When accessing the resource through LDAP, use the `ds-entry-checksum` attribute instead.

To Configure the SCIM Servlet Extension using the Batch Script

The following example procedure assumes that you have set up your certificates, keystore, and truststore

1. Open the `<server-root>/config/scim-config-ds.dsconfig` script in a text editor.
2. For the optional elements (W3C common log, filtering by DN, and VLV Indexes, remove the comment (#) symbol on the `dsconfig` commands. Save the file when finished editing.
3. To enable the SCIM servlet extension, run the `dsconfig` batch file. Remember to include the bind parameters.

```
$ bin/dsconfig --batch-file config/scim-config-ds.dsconfig
```

SCIM Servlet Extension Authentication

The SCIM servlet supports authentication using either the HTTP Basic authentication scheme, or OAuth 2.0 bearer tokens. When authenticating using HTTP Basic authentication, the SCIM servlet attempts to correlate the username component of the Authorization header to a DN in the Directory Server. If the username value cannot be parsed directly as a DN, it is correlated to a DN using an Identity Mapper. The DN is then used in a simple bind request to verify the password.

In deployments that use an OAuth authorization server, the SCIM extension can be configured to authenticate requests using OAuth bearer tokens. The SCIM extension supports authentication with OAuth 2.0 bearer tokens (per RFC 6750) using an OAuth Token Handler Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, PingDirectory Server provides the token handler API to decode incoming bearer tokens and extract or correlate associated authorization DNs.

Neither HTTP Basic authentication nor OAuth 2.0 bearer token authentication are secure unless SSL is used to encrypt the HTTP traffic.

To Configure Basic Authentication Using an Identity Mapper

By default, the SCIM servlet is configured to use the Exact Match Identity Mapper, which matches against the `uid` attribute. In this example, an alternate Identity Mapper is created so that clients can authenticate using `cn` values.

1. Create a new Identity Mapper that uses a match attribute of cn.

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "CN Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:cn
```

2. Configure the SCIM servlet to use the new Identity Mapper.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set "identity-mapper:CN Identity Mapper"
```

To Enable OAuth Authentication

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the Server SDK documentation.

1. Install your OAuth token handler on the server using `dsconfig`.

```
$ bin/dsconfig create-oauth-token-handler \
  --handler-name ExampleOAuthTokenHandler \
  --type third-party \
  --set extension-
class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
```

2. Configure the SCIM servlet extension to use it as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set oauth-token-handler:ExampleOAuthTokenHandler
```

Verifying the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

To Verify the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

- Run `curl` to verify that the SCIM extension is running. The -k (or --insecure) option is used to turn off curl's verification of the server certificate, since the example Directory Server is using a self-signed certificate.

```
$ curl -u "cn=Directory Manager:password" \
-k "https://localhost:8443/scim/ServiceProviderConfigs"

{"schemas":["urn:scim:schemas:core:1.0"],"id":"urn:scim:schemas:core:1.0",
"patch":{"supported":true},"bulk":{"supported":true,"maxOperations":10000,
"maxPayloadSize":10485760},"filter":{"supported":true,"maxResults":100},
"changePassword":{"supported":true},"sort":{"supported":true},
"etag":{"supported":false},"authenticationSchemes":[{"name":"HttpBasic",
"description":"The HTTP Basic Access Authentication scheme. This scheme is
not considered to be a secure method of user authentication (unless used in
conjunction with some external secure system such as SSL), as the user
name and password are passed over the network as cleartext.","specUrl":
"http://www.ietf.org/rfc/rfc2617","documentationUrl":
```

```
"http://en.wikipedia.org/wiki/Basic_access_authentication"}]}
```

- If the user ID is a valid DN (such as `cn=Directory Manager`), the SCIM extension authenticates by binding to the Directory Server as that user. If the user ID is not a valid DN, the SCIM extension searches for an entry with that `uid` value, and binds to the server as that user. To verify authentication to the server as the user with the `uid` of `user.0`, run the following command:

```
$ curl -u "user.0:password" \
  -k "https://localhost:8443/scim/ServiceProviderConfigs"
```

Configuring Advanced SCIM Extension Features

The following sections show how to configure advanced SCIM servlet extension features, such as bulk operation implementation, mapping SCIM resource IDs, and transformations.

Managing the SCIM Schema

This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About SCIM Schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping LDAP Schema to SCIM Resource Schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

Note that LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). The easiest and most-correct way to handle this is to make sure that any attributes that may contain binary data are declared using "dataType=binary" in the `scim-resources.xml` file. Likewise, when using the Identity Access API make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes which may contain binary data. This will cause the server to automatically base64-encode the data before returning it to clients and will also make it predictable for clients because they can assume the data will always be base64-encoded.

However, it is still possible that attributes that are not declared as binary in the schema may contain binary data (or just data that is invalid in XML), and the server will always check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values which are found to include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the <resource> Element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A resource element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the <attribute> Element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as urn:scim:schemas:core:1.0. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is false.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:

  `simple`: specifies a simple, singular SCIM attribute.
  `complex`: specifies a complex, singular SCIM attribute.
  `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
  `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the <simple> Element

A `simple` element has the following XML attributes:

- dataType: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: binary, boolean, dateTime, decimal, integer, string.
- caseExact: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simple element contains the following XML element:

- mapping: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the <complex> Element

The complex element does not have any XML attributes. It contains the following XML element:

- subAttribute: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard type, primary, and display sub-attributes do not need to be specified.

About the <simpleMultivalued> Element

A simpleMultiValued element has the following XML attributes:

- childName: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- dataType: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: binary, boolean, dateTime, integer, string.
- caseExact: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simpleMultiValued element contains the following XML elements in sequence:

- canonicalValue: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- mapping: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the <complexMultiValued> Element

A complexMultiValued element has the following XML attribute:

- tag: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is address.

A complexMultiValued element contains the following XML elements in sequence:

- subAttribute: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.
- canonicalValue: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the <subAttribute> Element

A subAttribute element has the following XML attributes:

- name: a required element specifying the name of the sub-attribute.
- readOnly: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- required: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- dataType: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: binary, boolean, dateTime, integer, string.
- caseExact: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the complexMultiValued element.

About the <canonicalValue> Element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the <mapping> Element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in the *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API* section.

About the <subMapping> Element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. The available transformations are described in *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API*.

About the <LDAPSearch> Element

An `LDAPSearch` element contains the following XML elements in sequence:

- `baseDN`: a required element specifying one or more LDAP search base DNs to be used when querying for the SCIM resource.
- `filter`: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- `resourceIDMapping`: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN. Note The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

Note: The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

About the <resourceIDMapping> Element

The `resourceIDMapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `scim-consumer`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or Directory

Server, meaning that a value is automatically provided by the Directory Server (as would be the case if the mapped LDAP attribute is entryUUID).

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the <LDAPAdd> Element

An LDAPAdd element contains the following XML elements in sequence:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

About the <fixedAttribute> Element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The value merge indicates that the fixed values should be merged with the existing values. The value overwrite indicates that the existing values are to be overwritten by the fixed values. The value preserve indicates that no changes should be made. The default value is merge.

A `fixedAttribute` element contains one or more `fixedValue` XML element, which specify the fixed LDAP values.

Validating Updated SCIM Schema

The PingDirectory Server SCIM extension is bundled with an XML Schema document, `resources.xsd`, which describes the structure of a `scim-resources.xml` resource configuration file. After updating the resource configuration file, you should confirm that its contents are well-formed and valid using a tool such as `xmllint`.

For example, you could validate your updated file as follows:

```
$ xmllint --noout --schema resources.xsd scim-resources.xml
scim-resources.xml validates
```

Mapping SCIM Resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. The `entryUUID` attribute, whose read-only value is assigned by the Directory Server, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership. The resource configuration allows the SCIM resource ID to be mapped to the LDAP entry DN. However, the entry DN does not meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN. Likewise, you can use the Identity Access API to change the value of an entry's RDN attribute, thereby triggering a MODDN operation.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Directory Server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin may be used by the Directory Server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see "Working with the UID Unique Attribute Plug-in" in the PingDirectory Server Administration Guide.

Note: Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the `<resourceIDMapping>` Element".

Using Pre-defined Transformations

Transformations are required to change SCIM data types to LDAP syntax values. The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation`. Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation`. Transforms SCIM dateTime data type values to LDAP Generalized Time syntax values and vice-versa.
- `com.unboundid.scim.ldap.PostalAddressTransformation`. Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the $ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.
- `com.unboundid.scim.ldap.TelephoneNumberTransformation`. Transforms LDAP Telephone Number syntax (E.123) to RFC3966 format and vice-versa.

You can also write your own transformations using the SCIM API described in the following section.

Mapping LDAP Entries to SCIM Using the SCIM-LDAP API

In addition to the SCIM SDK, PingDirectory Server provides a library called SCIM-LDAP, which provides facilities for writing custom transformations and more advanced mapping.

You can add the SCIM-LDAP library to your project using the following dependency:

```
<dependency>
    <groupId>com.unboundid.product.scim</groupId>
    <artifactId>scim-ldap</artifactId>
    <version>1.5.0</version>
</dependency>
```

Create your custom transformation by extending the `com.unboundid.scim.ldap.Transformation` class. Place your custom transformation class in a jar file in the server's `lib` directory.

Note: The Identity Access API automatically maps LDAP attribute syntaxes to the appropriate SCIM attribute types. For example, an LDAP DirectoryString is automatically mapped to a SCIM string.

SCIM Authentication

SCIM requests to the LDAP endpoints will support HTTP Basic Authentication and OAuth2 Authentication using a bearer token. There is existing support for this feature in the Directory Server and the Directory Proxy Server using the OAuthTokenHandler API (i.e., via a Server SDK extension, which requires some technical work to implement).

Note that our implementation only supports the HTTP Authorization header for this purpose; we do not support the form-encoded body parameter or URI query parameter mechanisms for specifying the credentials or bearer token.

SCIM Logging

The Directory Server already provides a detailed HTTP log publisher to capture the SCIM and HTTP request details. To be able to correlate this data to the internal LDAP operations that are invoked behind the scenes, the Access Log Publisher will use "`origin=scim`" in access log messages that are generated by the SCIM servlet.

For example, you will see a message for operations invoked by replication:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191
origin="replication" dn="uid=user.3,ou=people,dc=example,dc=com"
```

Likewise for SCIM messages, you will see a message like this:

```
[30/Oct/2012:18:45:10.490 -0500] MODFIY REQUEST conn=-3 op=190 msgID=191
origin="scim" dn="uid=user.3,ou=people,dc=example,dc=com"
```

SCIM Monitoring

There are two facilities that can be used to monitor the SCIM activity in the server.

- **HTTPConnectionHandlerStatisticsMonitorProvider** -- Provides statistics straight about total and average active connections, requests per connection, connection duration, processing time, invocation count, etc.
- **SCIMServletMonitorProvider** -- Provides high level statistics about request methods (POST, PUT, GET, etc.), content types (JSON, XML), and response codes, for example, "`user-patch-404:26`".

The LDAP object class endpoints are treated as their own resource types, so that for requests using the Identity Access API, there will be statistics, such as `person-get-200` and `inetorgperson-post-401`.

Configuring the Identity Access API

Once you have run the `<server-root>/config/scim-config-ds.dsconfig` script, the resources defined in the `scim-resources.xml` will be available as well as the Identity Access API. However, to allow SCIM access to the raw LDAP data, you must set a combination of configuration properties on the SCIM Servlet Extension using the `dsconfig` tool.

- **include-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will be exposed. The object class used here will be the one that clients need to use when referencing Identity Access API resources. This property allows the special value "*" to allow all object classes. If "*" is used, then the SCIM servlet uses the same case used in the Directory Server LDAP Schema.
- **exclude-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will not be exposed. When this property is specified, all object classes will be exposed except those in this list.
- **include-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will be exposed. If specified, only entries under these base DNs will be accessible. No parent-child relationships in the DNs are allowed here.
- **exclude-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will not be exposed. If specified, entries under these base DNs will not be accessible. No parent-child relationships in the DNs are allowed here.

Using a combination of these properties, SCIM endpoints will be available for all included object classes, just as if they were SCIM Resources defined in the `scim-resources.xml` file.

To Configure the Identity Access API

1. Ensure that you have run the `scim-config-ds.dsconfig` script to configure the SCIM interface. Be sure to enable the entryDN virtual attribute. See the Configure SCIM section for more information.
2. Set a combination of properties to allow the SCIM clients access to the raw LDAP data: `include-ldap-objectclass`, `exclude-ldap-objectclass`, `include-ldap-base-dn`, or `exclude-ldap-base-dn`.

   ```
   $ bin/dsconfig set-http-servlet-extension-prop \
     --extension-name SCIM --set 'include-ldap-objectclass:*' \
     --set include-ldap-base-dn:ou=People,dc=example,dc=com
   ```

   The SCIM clients now have access to the raw LDAP data via LDAP object class-based resources as well as core SCIM resources as defined in the `scim.resource.xml` file.

To Disable Core SCIM Resources

1. Open the `config/scim-resources.xml` file, and comment out or remove the `<resource>` elements that you would like to disable.

2. Disable and re-enable the HTTP Connection Handler, or restart the server to make the changes take effect. In general, changing the `scim-resources.xml` file requires a HTTP Connection Handler restart or server restart.

   Note: When making other changes to the SCIM configuration by modifying the SCIM HTTP Servlet Extension using `dsconfig`, the changes take effect immediately without any restart required.

To Verify the Identity Access API Configuration

- Perform a curl request to verify the Identity Access API configuration.

```
$ curl -k -u "cn=directory manager:password" \
  -H "Accept: application/json" \
  "https://example.com/top/56c9fd6b-f870-35ef-9959-691c783b7318?
     attributes=entryDN,uid,givenName,sn,entryUUID"
     {"schemas":
["urn:scim:schemas:core:1.0","urn:unboundid:schemas:scim:ldap:1.0"],
      "id":"56c9fd6b-f870-35ef-9959-691c783b7318",
      "meta":{"lastModified":"2013-01-11T23:38:26.489Z",
      "location":"https://example.com:443/v1/top/56c9fd6b-
f870-35ef-9959-691c783b7318"},
      "urn:unboundid:schemas:scim:ldap:1.0":{"givenName":["Rufus"],"uid":
["user.1"],
      "sn":["Firefly"],"entryUUID":["56c9fd6b-f870-35ef-9959-691c783b7318"],
      "entrydn":"uid=user.1,ou=people,dc=example,dc=com"}}
```

Monitoring the SCIM Servlet Extension

The SCIM SDK provides a command-line tool, `scim-query-rate`, that measures the SCIM query performance for your extension. The SCIM extension also exposes monitoring information for each SCIM resource, such as the number of successful operations per request, the number of failed operations per request, the number of operations with XML or JSON to and from the client. Finally, the Directory Server automatically logs SCIM-initiated LDAP operations to the default File-based Access Logger. These operations will have an `origin='scim'` attribute to distinguish them from operations initiated by LDAP clients. You can also create custom logger or request criteria objects that can track incoming HTTP requests, which the SCIM extension rewrites as internal LDAP operations.

Testing SCIM Query Performance

You can use the `scim-query-rate` tool, provided in the SCIM SDK, to test query performance, by performing repeated resource queries against the SCIM server.

The `scim-query-rate` tool performs searches using a query filter or can request resources by ID. For example, you can test performance by using a filter to query randomly across a set of one million users with eight concurrent threads. The user resources returned to the client in this example is in XML format and includes the `userName` and `name` attributes.

```
scim-query-rate --hostname server.example.com --port 80 \
--authID admin --authPassword password --xml \
--filter 'userName eq "user.[1-1000000]"' --attribute userName \
--attribute name --numThreads 8
```

You can request resources by specifying a resource ID pattern using the --resourceID argument as follows:

```
scim-query-rate --hostname server.example.com --port 443 \
--authID admin --authPassword password --useSSL --trustAll\
--resourceName User \
--resourceID 'uid=user.[1-150000],ou=people,dc=example,dc=com'
```

The `scim-query-rate` tool reports the error "`java.net.SocketException: Too many open files`" if the open file limit is too low. You can increase the open file limit to increase the number of file descriptors.

Monitoring Resources Using the SCIM Extension

The monitor provider exposes the following information for each resource:

Number of successful operations per request type (such as GET, PUT, and POST).
Number of failed operations and their error codes per request type.
Number of operations with XML or JSON from client.
Number of operations that sent XML or JSON to client.

In addition to the information about the user-defined resources, monitoring information is also generated for the schema, service provider configuration, and monitor resources. The attributes of the monitor entry are formatted as follows:

```
{resource name}-resource-{request type}-{successful or error status code}
```

You can search for one of these monitor providers using an `ldapsearch` such as the following:

```
$ bin/ldapsearch --port 1389 bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN cn=monitor \
  --searchScope sub "(objectclass=scim-servlet-monitor-entry)"
```

For example, the following monitor output was produced by a test environment with three distinct SCIM servlet instances, Aleph, Beth, and Gimel. Note that the first instance has a custom resource type called host.

```
$ bin/ldapsearch --baseDN cn=monitor \
  '(objectClass=scim-servlet-monitor-entry)'
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Aleph)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Aleph)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
user-resource-delete-successful: 1
user-resource-put-content-xml: 27
user-resource-query-response-json: 3229836
user-resource-put-403: 5
user-resource-put-content-json: 2
user-resource-get-401: 1
user-resource-put-response-json: 23
user-resource-get-response-json: 5
user-resource-get-response-xml: 7
user-resource-put-400: 2
user-resource-query-401: 1141028
user-resource-post-content-json: 1
user-resource-put-successful: 22
user-resource-post-successful: 1
user-resource-delete-404: 1
user-resource-query-successful: 2088808
```

```
user-resource-get-successful: 10
user-resource-put-response-xml: 6
user-resource-get-404: 1
user-resource-delete-401: 1
user-resource-post-response-json: 1
host-resource-query-successful: 5773268
host-resource-query-response-json: 11576313
host-resource-query-400: 3
host-resource-query-response-xml: 5
host-resource-query-401: 5788152
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Beth)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Beth)
version: 1.2.0
build: 20120105174457Z
revision: 820
serviceproviderconfig-resource-get-successful: 3
serviceproviderconfig-resource-get-response-json: 2
serviceproviderconfig-resource-get-response-xml: 1
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
group-resource-query-successful: 245214
group-resource-query-response-json: 517841
group-resource-query-400: 13711
group-resource-query-401: 258916
user-resource-query-response-json: 107876
user-resource-query-400: 8288
user-resource-get-400: 33
user-resource-get-response-json: 1041
user-resource-get-successful: 2011
user-resource-query-successful: 45650
user-resource-get-response-xml: 1003
user-resource-query-401: 53938
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Gimel)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Gimel)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 1
schema-resource-query-401: 1
schema-resource-query-response-json: 2
user-resource-query-successful: 65
user-resource-get-successful: 4
user-resource-get-response-json: 6
user-resource-query-response-json: 132
user-resource-get-404: 2
```

```
user-resource-query-401: 67
```

About the HTTP Log Publishers

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.
- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.
- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as "'['dd/MM/yyyy:HH:mm:ss Z']'"
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information than the common log format in a format that is familiar to administrators who use the File-Based Access Log Publisher.

The HTTP Detailed Access Log Publisher generates log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-
38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-
d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the `dsconfig` command-line tool help.

Chapter 24

Managing Server SDK Extensions

Topics:

- *About the Server SDK*
- *Available Types of Extensions*

The PingDirectory Server provides support for any custom extensions that you create using the Server SDK. This chapter summarizes the various features and extensions that can be developed using the Server SDK.

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Available Types of Extensions

The Server SDK provides support for creating a number of different types of extensions for Ping Identity Server Products, including the PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. Some of those extensions include:

Cross-Product Extensions

Access Loggers
Alert Handlers
Error Loggers
Key Manager Providers
Monitor Providers
Trust Manager Providers
OAuth Token Handlers
Manage Extension Plugins

PingDirectory Server Extensions

Certificate Mappers
Change Subscription Handlers
Extended Operation Handlers
Identity Mappers
Password Generators
Password Storage Schemes
Password Validators
Plugins
Tasks
Virtual Attribute Providers

PingDirectoryProxy Server Extensions

LDAP Health Checks
Placement Algorithms
Proxy Transformations

PingDataSync Server Extensions

JDBC Sync Sources
JDBC Sync Destinations
LDAP Sync Source Plugins
LDAP Sync Destination Plugins
Sync SourcesSync Destinations

- Sync Pipe Plugins

For more information on the Server SDK, see the documentation available in the SDK build.

Chapter 25

Troubleshooting the Server

Topics:

The PingDirectory Server provides a highly-reliable service that satisfies your company's objectives. However, if problems do arise (whether from issues in the Directory Server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

This chapter provides information about how to perform this analysis to help ensure that the problem is resolved as quickly as possible. This chapter presents the following information:

Working with the Collect Support Data Tool

The Directory Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the `collect-support-data` tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool varies between systems. However, the tool always tries to get the same information across all systems for the target Directory Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Server Commands Used in the Collect Support Data Tool

The following presents a summary of the data collectors that the `collect-support-data` tool archives in zip format. If an error occurs during processing, you can re-run the specific data collector command and send the results to your authorized support provider.

Table 66: Directory Server Commands Used in the Collect-Support-Data Tool

Data Collector	Description
status	Runs `status` -F to show the full version information of the Directory Server (Unix, Windows).
server-state	Runs `server-state` to show the current state of the Directory Server process (Unix, Windows).
dsreplication status	Runs `dsreplication status` to show the status of the replicated topology (Unix, Windows). If the --noReplicationStatus option is used, the replication status information is not collected.

JDK Commands Used in the Collect-Support-Data Tool

Table 67: JDK Commands Used in the Collect-Support-Data Tool

Data Collector	Description
jps	Java Virtual Machine Process status tool. Reports information on the JVM (Linux, Windows, Mac OS).
jstack	Java Virtual Machine Stack Trace. Prints the stack traces of threads for the Java process (Linux, Windows, Mac OS).
jstat	Java Virtual Machine Statistics Monitoring Tool. Displays performance statistics for the JVM (Linux, Windows, Mac OS).
jinfo	Displays the Java configuration information for the Java process (Linux, Windows, Mac OS).

Linux Commands Used in the collect-support-data Tool

Table 68: Linux Commands Used in the Collect-Support-Data Tool

Data Collector	Description
tail	Displays the last few lines of a file. Tails the `/var/logs/messages` directory.
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
cat	Concatenates the following files and prints to standard output: /proc/cpuinfo /proc/meminfo /etc/hosts /etc/nsswitch.conf /etc/resolv.conf
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active processed specified by the pid.
top	Prints a list of active processes and how much CPU and memory each process is using.

MacOS Commands Used in the Collect Support Data Tool

Table 69: MacOS Commands Used in the Collect-Support-Data Tool

Data Collector	Description
uname	Prints system, machine, and operating system information.
uptime	Prints the time the server has been up and active.
ps	Prints a snapshot of the current active processes.
system_profiler	Prints system hardware and software configuration.
vm_stat	Prints machine virtual memory statistics.
tail	Displays the last few lines of a file. Tails the `/var/log/system.log` directory.
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
sample	Profiles a process during an interval.

Available Tool Options

The `collect-support-data` tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.
- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.
- **--sequential**. Use this option to diagnose “Out of Memory” errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.
- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, `vmstat`, `iostat`, or `mpstat`). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.
- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, `mpstat`). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.
- **--maxJstacks {number}**. Specifies the number of jstack samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they will impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. The comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain `dsreplication status` information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain `dsreplication status` information.

To Run the Collect Support Data Tool

1. Go to the server root directory.
2. Use the `collect-support-data` tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --serverRoot /opt/PingDirectory --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Directory Server Troubleshooting Information

The Directory Server has a comprehensive default set of log files and monitor entries that are useful when troubleshooting a particular server problem.

Error Log

By default, this log file is available at `logs/errors` below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE msgID=458887
 msg="The Directory Server has started successfully"
```

The category field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL, ADMIN, ADMIN_TOOL, BACKEND, CONFIG, CORE, DSCONFIG, EXTENSIONS, PROTOCOL, SCHEMA, JEB, SYNC, LOG, PLUGIN, PROXY, QUICKSETUP, REPLICATION, RUNTIME_INFORMATION, TASK, THIRD_PARTY, TOOLS, USER_DEFINED, UTIL, VERSION.

The severity field provides information about how severe the server considers the problem to be. Available severities include:

- **DEBUG** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Server provides a separate debug logging facility as described below.
- **INFORMATION** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.
- **MILD_WARNING** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.
- **MILD_ERROR** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.
- **NOTICE** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.
- **SEVERE_WARNING** – Used for problems that the server detects that might lead to bigger problems in the future and should be addressed by administrators.
- **SEVERE_ERROR** – Used for significant problems that have prevented the server from successfully completing processing and are considered important.
- **FATAL_ERROR** – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the NOTICE, SEVERE_WARNING, SEVERE_ERROR, and FATAL_ERROR severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the PROTOCOL category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-
warning,severe-error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the DEBUG severity for a category will not automatically enable logging at the INFORMATION, MILD_WARNING, or MILD_ERROR severities.

The error log configuration may be altered on the fly using tools like `dsconfig`, the Administrative Console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out Log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. The `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug Log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- **enabled** – Indicates whether debug logging is enabled. By default, it is disabled.
- **log-file** – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- **default-debug-level** – Specifies the minimum log level for debug messages that should be written. The default value is “error,” which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include warning, info, and verbose. Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the info debug level automatically enables the warning and error levels.
- **default-debug-category** – Specifies the categories for debug messages that should be written. Some of the most useful categories include caught (provides information and stack traces for any exceptions caught during processing), database-access (provides information about operations performed in the underlying database), protocol (provides information about ASN.1 and LDAP communication performed by the server), and data (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files to help isolate information that might be relevant to a particular problem.

Note: Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and then be scoped so that only pertinent debug information is recorded.

Debug targets can be used to further pare down the set of messages generated. For example, you can specify that the debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and interpret the output.

Replication Repair Log

The replication repair log is written to `logs/replication` by default and records information about processing performed by the replication repair service. This log is used to resolve replication conflicts that can arise. For example, if the same entry is modified at the same time on two different systems, or if an attempt is made to create entries with the same DN at the same time on two different systems, the Directory Server records these events.

Config Audit Log and the Configuration Archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the `logs/config-audit.log` file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the `dsconfig` command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.

- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Server maintains complete copies of all previous configurations. These configurations are provided in the `config/archived-configs` directory and are gzip-compressed copies of the `config/config.ldif` file in use before the configuration change was made. The filenames contain time stamps that indicate when that configuration was first used.

Access and Audit Log

The access log provides information about operations processed within the server. The default access log file is written to `logs/access`, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1"
 to="127.0.0.1"
  protocol="LDAP+TLS" clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
  dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
 clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2
  base="ou=People,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL"
  resultCode=0 etime=0.549 entriesReturned=1
[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensi- tive or not useful.

Result messages include a `resultCode` element that indicates whether the operation was successful or if failed and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **origin=replication** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **message** – Text that was included in the `diagnosticMessage` field of the response sent to the client.
- **additionalInfo** – Additional information about the operation that was not included in the response sent back to the client.
- **authDN – DN** of the user that authenticated to the server (typically only included in bind result messages).
- **authzDN – DN** of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **authFailureID** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).
- **authFailureReason** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **responseOID** – OID included in an extended response returned to the client.
- **entriesReturned** – Number of matching entries returned to the client for a search operation.

- **unindexed=true** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The LDAP SDK for Java provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify DN) processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to `logs/audit`) in LDIF form.

The PingDirectory Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup Log

The `setup` tool writes a log file providing information about the processing that it performs. By default, this log file is written to `logs/setup.log` although a different name may be used if a file with that name already exists, because the `setup` tool has already been run. The full path to the setup log file is provided when the `setup` tool has completed.

Tool Log

Many of the administrative tools provided with the Directory Server (for example, `import-ldif`, `export-ldif`, `backup`, `restore`, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the `logs/tools` directory (for example, **logs/tools/ import-ldif.log**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

je.info and je.config Files

The primary data store used by the Directory Server is the Oracle Berkeley DB Java Edition (JE). The Directory Server provides two primary sources of information about processing within the database.

The first is logging performed by the JE code itself, and is written into the `je.info.0` file in the server containing the database files (for example, `db/userRoot/je.info.0`). In the event of a problem within JE itself, useful information about the nature of the problem may be written to this log. The level of information written to this log file is controlled by the `db-logging-level` property in the backend configuration object. It uses the standard Java logging framework for logging messages, so the standard SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST levels are available.

The second is configuration information used when opening the database environment. When the backend database environment is opened, then the Directory Server will also write a file named `je.config` in the server containing the database files (for example, `db/userRoot/je.config`) with information about the configuration used.

LDAP SDK Debug Log

This log can be used to help examine the communication between the Directory Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

About the Monitor Entries

While the Directory Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the `cn=monitor` subtree. The types of monitor entries that are available include:

- **General Monitor Entry (cn=monitor)** – Provides a basic set of general information about the server.
- **Active Operations Monitor Entry (cn=Active Operations,cn=monitor)** – Provides information about all operations currently in progress in the server.
- **Backend Monitor Entries (cn={id} Backend,cn=monitor)** – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- **Client Connections Monitor Entry (cn=Client Connections,cn=monitor)** – Provides information about all connections currently established to the server.
- **Connection Handler Monitor Entry (cn={name},cn=monitor)** – Provides information about the configuration of each connection handler and the client connections established to it.
- **Database Environment Monitor Entries (cn={id} Database Environment,cn=monitor)** – Provides statistics and other data from the Oracle Berkeley DB Java Edition database envi- ronment used by the associated backend.
- **Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor)** – Provides infor- mation about the amount of usable disk space available to server components.
- **JVM Memory Usage Monitor Entry (cn=JVM Memory Usage,cn=monitor)** – Provides information about garbage collection activity, the amount of memory available to the server, and the amount of memory consumed by various server components.
- **JVM Stack Trace Monitor Entry (cn=JVM Stack Trace,cn=monitor)** – Provides a stack trace of all threads in the JVM.
- **LDAP Statistics Monitor Entries (cn={name} Statistics,cn=monitor)** – Provides information about the number of each type of operation requested and bytes transferred over the connection handler.
- **Processing Time Histogram Monitor Entry (cn=Processing Time Histogram,cn=monitor)** – Provides information about the number of percent of operations that completed in various response time categories.
- **SSL Context Monitor Entry (cn=SSL Context,cn=monitor)** – Provides information about the available and supported SSL Cipher Suites and Protocols on the server.
- **System Information Monitor Entry (cn=System Information,cn=monitor)** – Provides information about the underlying JVM and system.
- **Version Monitor Entry (cn=Version,cn=monitor)** – Provides information about the Directory Server version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Directory Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Directory Server Troubleshooting Tools

The PingDirectory Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server Version Information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Server software that is in use. If the server is running, then this information can be obtained from the "`cn=Version,cn=monitor`" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.

- Source revision number from which the server was built.
- Build information including build ID with time stamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Jetty, JZlib, SNMP4j (SNMP4J, Agent, Agentx), Groovy, LDAP SDK for Java, and the Server SDK.

LDIF Connection Handler

The Directory Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This Directory Server does not exist by default, but if it is created and an LDIF file is placed in it that contains one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

dbtest Tool

The `dbtest` tool provides a utility that can be used to obtain general information about the data in a backend database. The tool dumps information about entries or keys, and raw data from the database. It can also find keys that have exceeded the entry limit.

For example, the following command can be used to dump a list of all keys in the `objectClass.equality` that have exceeded the entry threshold:

```
$ bin/dbtest dump-database-container \
  --backendID userRoot \
  --baseDN "dc=example,dc=com" \
  --databaseName objectClass.equality \
  --onlyExceedingLimit
```

On a large database, many `dbtest` operations may take a long time to complete, since every record in the associated database is examined. Use the database name option to list a specific database. The following command displays information about the uid.equality database in the `dc=example,dc=com` entry container in the userRoot backend.

```
$ bin/dbtest list-database-containers -n userRoot -b "dc=example,dc=com" -d
 uid.equality
```

Index Key Entry Limit

Indexes have keys that maintain a list of matching entries, up to the index entry limit. When that limit is reached, the key will not contain or maintain that list, and will just maintain a count of matching entries. To determine if index keys are approaching their limit, use either the `dbtest` tool or the `verify-index` tool.

While the `dbtest` tool can be used to gather general imformation, the `verify-index` tool provides statistical data about the percent of entries covered by the keys.

For example, the following command can be used to retrieve a list of keys that have exceeded the entry threshold:

```
$ bin/verify-index \
  --baseDN dc=example,dc=com \
  --listKeysExceedingIndexEntryLimit
```

The following is a sample of the data returned:

```
 [12:06:05]  Checked 6003 entries and found 0 error(s) in 2 seconds (average
 rate 2453.2/sec)
```

```
[12:06:05]  Statistics for records that have exceeded the entry limit:
[12:06:05]  The st.equality index has 48 such record(s) limit=100 min=103
max=152 median=118
[12:06:05]  1. or (152 entries / 2.53% of all entries)
[12:06:05]  2. ma (132 entries / 2.20% of all entries)
 .....
[12:06:05]  The id2subtree index has 2 such record(s) limit=4000 min=6000
max=6002 median=6001
[12:06:05]  1. 1 => dc=example,dc=com (6002 entries / 99.98% of all entries)
 .....
[12:06:05]  The id2children index has 1 such record(s) limit=4000 min=6000
max=6000 median=6000
[12:06:05]  1. 2 => ou=People,dc=example,dc=com (6000 entries / 99.95% of all
entries)
[12:06:05]  The objectClass.equality index has 4 such record(s) limit=4000
min=6001 max=6003 median=6001
[12:06:05]  1. top (6003 entries / 100.00% of all entries)
 .....
```

Embedded Profiler

If the Directory Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the `dsconfig` tool or the Administrative Console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The `profile-action` property for this configuration object can have one of the following values:

- **start** – Indicates that the embedded profiler should start capturing data in the background.
- **stop** – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a `logs/profile{timestamp}` file.
- **cancel** – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the `profiler-viewer` tool. This tool can operate in either a text-based mode, in which case it dumps a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

To Invoke the Profile Viewer in Text-based Mode

- Run the `profile-viewer` command and specify the captured log file using the --fileName option.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

To Invoke the Profile Viewer in GUI Mode

- Run the `profile-viewer` command and specify the captured log file using the --fileName option. To invoke GUI mode, add the option --useGUI.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Oracle Berkeley DB Java Edition Utilities

The Oracle Berkeley DB Java Edition (JE) itself provides a number of utilities that can be used for performing various types of low-level debugging in the database environment. These utilities should generally not be used unless you are advised to do so by your authorized support provider, but they provide access to information about the underlying database environment that is not available through any other means.

Troubleshooting Resources for Java Applications

Because the PingDirectory Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Server. In many cases, obtaining the full benefit of these tools requires access to the Directory Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java Troubleshooting Tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the PingDirectory Server.

jps

The `jps` tool is a Java-specific version of the UNIX `ps` tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- **-v** – Includes the arguments passed to the JVM for the processes that are listed.
- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

jstack

The `jstack` tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can have user-friendly names (as do the threads used by the PingDirectory Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where {processID} is the process ID of the target JVM as returned by the `jps` command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the java command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The "-l" option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The “-m” option can be used to include native frames in the stack trace.

jmap

The jmap tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native pmap tool provided by many operating systems. As with the jstack tool, jmap can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the jhat tool
- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.
- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the ":live" portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

jhat

The jhat (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory Server encounters an out of memory error (as a result of the "-XX:+HeapDumpOnOutOfMemoryError" JVM option) or from the use of the jmap command with the "-dump" option.

The jhat tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The jhat tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the jhat web server to begin listening on port 7000. It can be accessed in a browser at http://localhost:7000 (or http://address:7000 from a remote system). An alternate port number can be specified using the "-port" option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL http://localhost:7000/oql/ (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at http://localhost:7000/oqlhelp/.

jstat

The jstat tool is used to obtain a variety of statistical information from the JVM, much like the vmstat utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The {interval} option specifies the length of time in milliseconds between lines of output. The {processID} option specifies the process ID of the JVM used to run the Directory Server, which can be obtained by running jps as mentioned previously. The {type} option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.
- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.

- **-gccapacity** – Provides information about memory region capacities.

Java Diagnostic Information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

JVM Crash Diagnostic Information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the base directory of the Directory Server installation. This file includes information on the underlying cause of the JVM crash, information about the threads running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system.

Troubleshooting Resources in the Operating System

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the Directory Server, and in others system, tools can help narrow down the cause of the problem.

Identifying Problems with the Underlying System

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. To look for problems in the underlying system view the system log file (`/var/log/messages` on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.

Examining CPU Utilization

Observing CPU utilization for the Directory Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU Utilization

To investigate CPU consumption of the system as a whole, use the `vmstat` command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU Utilization

To investigate CPU consumption on a per-CPU basis, use the `mpstat` command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "-P ALL" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but mpstat reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-Process Utilization

To investigate CPU consumption on a per-process basis, use a command such as the top utility on Linux. If a process other than the Java process used to run the Directory Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Server to run effectively.

Examining Disk Utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The iostat tool may be used to obtain information about the disk activity on the system.

On Linux systems, iostat should be invoked with the "-x" argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but to obtain an initial feel for how busy the underlying disks are, look at the "%util" column on Linux. This field shows the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a tmpfs filesystem).

Examining Process Details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard ps tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard ps command displays only a limited set of the arguments based on available space in the terminal window.

pstack

The pstack command can be used to obtain a native stack trace of all threads in a process. While a native stack trace might not be as user-friendly as a Java stack trace obtained using jstack, it includes threads that are not available in a Java stack trace. For example, the command displays those threads used to perform garbage collection and other housekeeping tasks. The general usage for the pstack command is:

```
pstack {processID}
```

dbx / gdb

A process debugger provides the ability to examine a process in detail. Like `pstack`, a debugger can obtain a stack trace for all threads in the process, but it also provides the ability to examine a process (or core file) in much greater detail, including observing the contents of memory at a specified address and the values of CPU registers in different frames of execution. The GNU debugger `gdb` is widely-used on Linux systems.

Note that using a debugger against a live process interrupts that process and suspends its execution until it detaches from the process. In addition, when running against a live process, a debugger has the ability to actually alter the contents of the memory associated with that process, which can have adverse effects. As a result, it is recommended that the use of a process debugger be restricted to core files and only used to examine live processes under the direction of your authorized support provider.

pfiles / lsof

To examine the set of files that a process is using (including special types of files, like sockets), you can use a tool such as `lsof` on Linux systems, (

```
lsof -p {processID}
```

)

Tracing Process Execution

If a process is unresponsive but is consuming a nontrivial amount of CPU time, or if a process is consuming significantly more CPU time than is expected, it might be useful to examine the activity of that process in more detail than can be obtained using a point-in-time snapshot. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

This can be accomplished using the `strace` tool on Linux (

```
strace -f -p {processID}
```

).

Consult the `strace` manual page for additional information.

Problems with SSL Communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-server` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Examining Network Communication

Because the PingDirectory Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Server itself can provide details about its interaction with clients by enabling debugging for the protocol or data debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the *tcpdump* tool on Linux, can be used to accomplish this.

There are many options that can be used with these tools, and their corresponding manual pages will provide a more thorough explanation of their use. However, to perform basic tracing to show the full details of the packets received for communication on port 389 with remote host 1.2.3.4, the following command can be used on Linux:

```
tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

It does not appear that the `tcpdump` tool provides support for LDAP parsing. However, it is possible to write capture data to a file rather than displaying information on the terminal (using "-w {path}" with `tcpdump`), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access.

Common Problems and Potential Solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General Methodology to Troubleshoot a Problem

When a problem is detected, Ping Identity recommends using the following general methodology to isolate the problem:

1. Run the `bin/status` tool or look at the server status in the Administrative Console. The `status` tool provides a summary of the server’s current state with key metrics and a list of recent alerts.
2. Look in the server logs. In particular, view the following logs:

   logs/errors
   logs/failed-ops
   logs/expensive-ops
3. Use system commands, such as `vmstat` and `iostat` to determine if the server is bottle-necked on a system resource like CPU or disk throughput.
4. For performance problem (especially intermittent ones like spikes in response time), enabling the `periodic-stats-logger` can help to isolate problems, because it stores important server performance information on a per-second basis. The `periodic-stats-logger` can save the information in a csv-formatted file that can be loaded into a spreadsheet. The information this logger makes available is very configurable. You can create multiple loggers for different types of information or a different frequency of logging (for example, hourly data in addition to per-second data). For more information, see "Profiling Server Performance Using the Periodic Stats Logger".
5. For replication problem, run `dsreplication status` and look at the `logs/replication` file.
6. For more advanced users, run the `collect-support-data` tool on the system, unzip the archive somewhere, and look through the collected information. This is often useful when administrators most familiar with the PingData Platform do not have direct access to the systems where the production servers are running. They can examine the `collect-support-data` archive on a different server. For more information, see Using the Collect Support Data Tool.

Important: Run the `collect-support-data` tool whenever there is a problem whose cause is not easily identified, so that this information can be passed back to your authorized support provider before corrective action can be taken.

The Server Will Not Run Setup

If the `setup` tool does not run properly, some of the most common reasons include the following:

A Suitable Java Environment Is Not Available

The PingDirectory Server requires that Java be installed on the system and made available to the server, and it must be installed prior to running `setup`. If the `setup` tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the `setup` tool should be invoked with an explicitly-defined value for the *JAVA_HOME* environment variable that specifies the path to the Java installation that should be used. For example:

```
env JAVA_HOME=/ds/java ./setup
```

If this still does not work for some reason, then it can be that the value specified in the provided *JAVA_HOME* environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Oracle Berkeley DB Java Edition Is Not Available

If the version of the Directory Server that you are using was not provided with the Oracle Berkeley DB Java Edition library, then it must be manually downloaded and the appropriate JAR file placed in the `lib` directory before running setup. See the `lib/downloading-je.txt` file for instructions on obtaining the appropriate library.

Unexpected Arguments Provided to the JVM

If the `setup` script attempts to launch the java command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running setup, but this might not be the case if either the *JAVA_ARGS* or *UNBOUNDID_JAVA_ARGS* environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run `setup`:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Used

The `setup` tool is only intended to provide the initial configuration for the Directory Server. It refuses to run if it detects that the `setup` tool has already been run, or if an attempt has been made to start the Directory Server prior to running the `setup` tool. This protects an existing Directory Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run `setup` in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run `setup` failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the `setup` program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The Server Will Not Start

If the Directory Server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running

Only a single instance of the Directory Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other administrative operations can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Server could not acquire an exclusive lock on file
/ds/PingDirectory/locks/server.lock: The exclusive lock requested for file
/ds/PingDirectory/locks/ server.lock was not granted, which indicates
that another process already holds a shared or exclusive lock on that
file. This generally means that another instance of this server is already
running
```

If the Directory Server is not running (and is not in the process of starting up or shutting down) and there are no other tools running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the `locks` directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the filesystem.

There Is Not Enough Memory Available

When the Directory Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Server might need to be re-configured to use a smaller amount of memory than had been previously configured.
- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Server, or reconfigure the Directory Server to reduce the amount of memory that it attempts to use.
- The Directory Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the `vmstat` command and wait until the amount of free memory stops growing before attempting to restart the server.
- If the system is configured with one or more memory-backed filesystems, then look to see if there are any large files that can be consuming a significant amount of memory in any of those locations. If so, then remove them or relocate them to a disk-based filesystem.
- For Linux systems only, if there is a mismatch between the huge pages setting for the JVM and the huge pages reserved in the operating system.

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An Invalid Java Environment or JVM Option Was Used

If an attempt to start the Directory Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the `config/java.properties` file to reference to path to the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the

new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.

- If an *UNBOUNDID_JAVA_HOME* or *UNBOUNDID_JAVA_BIN* environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the `config/java.properties` file. Similarly, if an *UNBOUNDID_JAVA_ARGS* environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the *UNBOUNDID_JAVA_HOME*, *UNBOUNDID_JAVA_BIN*, and *UNBOUNDID_JAVA_ARGS* environment variables before trying to start the server.

Note that any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, then it can be necessary to remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option Was Provided

There are a small number of arguments that are provided when running the `bin/start-server` command, but in most cases, none are required. If one or more command-line arguments were provided for the `bin/start-server` command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration

If a change is made to the Directory Server configuration using an officially-supported tool like `dsconfig` or the Administrative Console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the `logs/config-audit.log` file to see what recent configuration changes have been made with the server online, or in the `config/archived-configs` directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the --useLastKnownGoodConfig option:

```
$ bin/start-server --useLastKnownGoodConfig
```

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the `config/config.ldif` file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.

Note: In addition to manually-editing the config file, you can look at previous achived configurations to see if the most recent one works. You can also use the `ldif-diff` tool to compare the configurations in the archive to the current configuration to see what is different.

You Do Not Have Sufficient Permissions

The Directory Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/PingDirectory
```

The Server Has Crashed or Shut Itself Down

You can first check the current server state by using the `bin/server-state` command. If the Directory Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using “kill -9”), then messages are written to the `error` and `server.out` logs explaining the reason for the shutdown.
- The Directory Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the `uptime` command on the underlying system shows that it was recently booted.
- The Directory Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the `error` and `server.out` logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Server was running crashed. If this happens, then the JVM should dump a fatal error log (a `hs_err_pid{processID}.log` file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

Conditions for Automatic Server Shutdown

All PingDirectory Server servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The Directory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
            --set unrecoverable-database-error-mode:initiate-server-shutdown
```

The Server Will Not Accept Client Connections

You can first check the current server state by using the `bin/server-state` command. If the Directory Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Server is not running.
- The underlying system on which the Directory Server is installed is not running.
- The Directory Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.

- If the Directory Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.
- If the Directory Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The Server is Unresponsive

You can first check the current server state by using the `bin/server-state` command. If the Directory Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the `jstack` command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

  A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

  Note: If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.
- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the `jstack` command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.
- If the JVM in which the Directory Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a `pstack` of the native process should show that most threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.
- If the JVM in which the Directory Server is running has hung for some reason, then the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like `snoop` or `tcpdump` should show that packets sent to the system on which the Directory Server is running are not receiving TCP acknowledgement.
- If the system on which the Directory Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is Slow to Respond to Client Requests

If the Directory Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see "Profiling Server Performance Using the Periodic Stats Logger". The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second. You can also run the `bin/status` command or view the status in the Administrative Console to see the Directory Server's Work Queue information (also see the next bullet point).
- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the `bin/status` command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

  ```
             --- Work Queue ---
             : Recent : Average : Maximum
  -----------:--------:---------:--------
  Queue Size : 10  : 1       : 10
  % Busy     : 17     : 14      : 100
  ```

  You can also view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second.
- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, `iostat` reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see Disk-Bound Deployments.
- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.
- If the Directory Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, `iostat` should report very high utilization for disks used to hold swap space, and commands like `cat /proc/meminfo` on Linux can report a large amount of swap memory in use. Another cause of swapping is if swappiness is not set to 0 on Linux. For more information, see Disable File System Swapping.
- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server Returns Error Responses to Client Requests

If a large number of client requests are receiving error responses, then view the `logs/failed-ops` log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.
- If a portion of the Directory Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxies or load balancers or both to route requests away from the affected server. As of Directory Server version 3.1 or later, the Directory Server will indicate that it is in a degraded status and the Directory Proxy Server will route around it.
- If the Directory Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.
- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

The Server Must Disconnect a Client Connection

If a client connection must be disconnected due to the expense of the client's request, such as an unindexed search across a very large database, perform the following:

- Find the client's connection ID by looking in the `cn=Active Operations,cn=monitor monitor` entry.

```
$ bin/ldapsearch -baseDN cn=monitor "cn=active operations" \
  --bindDN "cn=directory manager"  \
  --bindPassword password
```

- The monitor entry will contain attribute values for `operation-in-progress`, which look like an access log message. Look for the value of `conn` in the client request that should be disconnected. In the following example, the client to be disconnected is requesting a search for `(description=expensive)`, which is on connection 6.

```
dn: cn=Active Operations,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-active-operations-monitor-entry
objectClass: extensibleObject
cn: Active Operations
num-operations-in-progress: 2
operation-in-progress: [15/Dec/2014:10:55:35 -0600] SEARCH conn=6 op=3
 msgID=4
     clientIP="10.8.4.21" authDN="cn=app1,ou=applications,dc=example,dc=com"
 base="dc
     =example,dc=com" scope=wholeSubtree filter="(description=expensive)"
 attrs="A
     LL" unindexed=true
operation-in-progress: [15/Dec/2014:10:56:11 -0600] SEARCH conn=7 op=1
 msgID=2
     clientIP="127.0.0.1" authDN="cn=Directory Manager,cn=Root
 DNs,cn=config" base="c
     n=monitor" scope=wholeSubtree filter="(cn=active operations)"
 attrs="ALL"
     num-persistent-searches-in-progress: 0
```

- With the connection ID value, create a file with the following contents, named `disconnect6.ldif`.

```
dn: ds-task-id=disconnect6,cn=scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-disconnect
ds-task-disconnect-connection-id: 6
ds-task-id: disconnect6
ds-task-class-name:
 com.unboundid.directory.server.tasks.DisconnectClientTask
```

- This LDIF file represents a task entry. The connection ID value 6 is assigned to `ds-task-disconnect-connection-id`. The value for `ds-task-id` value does not follow a specific convention. It must be unique among other task entries currently cached by the server.
- Disconnect the client and cancel the associated operation by adding the task entry to the server:

```
$ bin/ldapmodify --filename disconnect6.ldif  \
  --defaultAdd --bindDN "cn=directory manager" \
  --bindPassword password
```

The Server is experiencing problems with replication

If replication does not appear to be functioning properly, then first check the `dsreplication status` command, which shows all of the servers that are replicating and whether they are back-logged or not. Next, you can check the server error log, replication repair log, and replication monitor entries may provide information about the nature of the underlying problem. Potential reasons that replication may not be functioning as expected include the following:

- Replication has not yet been configured between systems or has been disabled.
- If a server has been offline for a period of time or has fallen far enough behind such that it is missing changes, which are no longer present in any of the replication databases, then that server must be re-initialized with an up-to-date copy of the data from another server.
- If the environment is comprised of a heterogeneous set of systems, then it is possible that some of the systems might not be able to keep up with the maximum throughput achieved by other servers in the topology. In such cases, the slower servers might not be fast enough to remain in sync with the other servers in the environment.
- If the environment contains systems in multiple data centers and the network links between the data centers are insufficient for the volume of changes that must be processed, then servers might not be able to remain in sync under a high volume of changes.
- A network or firewall configuration problem has arisen, which prevents or interferes with communication between servers.
- An internal problem within the server has caused replication to stop functioning properly. The Directory Server logs the event in the error log in this case. Run the `collect-support-data` tool, so that the details of the problems can be passed to your authorized support provider. Then, try restarting the Directory Server.

How to Regenerate the Server ads-certificate

At setup time, the server generates a private key and certificate for use when secure communication between servers is required. This certificate, `ads-certificate`, is stored in `config/ads-truststore` and should typically remain unchanged for the life of the server deployment. If the need arises for a new ads-certificate to be created, say because the server-root has been copied to a new host, then the private key and certificate will be recreated by the startup process if the `config/ads-truststore` and `config/ads-truststore.pin` files are first manually removed while the server is offline. Note that if replication is enabled, the server must have replication disabled before regeneration of the ads-certificate.

For example, the Directory Server allows easy copying of its installation, which can then be used to install another server instance. If a server (ldap1.example.com:389) is enabled with its own copy (ldap2.example.com:389), `dsreplication` will exit with the following error message:

```
Replication cannot be enabled between servers ldap1.example.com:389 and
 ldap2.example.com:389
because they are using the same instance key.
```

The solution is to stop the server, remove `config/adstruststore` and `config/adstruststore.pin` and re-start the server. Upon startup, a new `adstruststore`, containing the server's instance key, will be generated. Then, you can re-run `dsreplication enable` to set up replication between the two servers.

The Server behaves differently from Sun/Oracle

After migrating from a Sun/Oracle configuration to a PingDirectory Server, follow the tuning procedures in *Sun/Oracle Compatibility* if the Directory Server behaves differently from the Sun/Oracle server.

Troubleshooting ACI Evaluation

The Directory Server provides the ability to collect debug information related to ACI evaluation for any operation by enabling the Debug ACI Logger. The Debug ACI Logger is highly configurable and can be scoped to trace very specific request operations in order to narrow on any ACI issue that may arise in the field. Parameters for non-request operations, such as `log-connects`, `log-disconnects`, `log-security-negotiation`, `log-results`, `log-assuance-completed`, `log-search-entries`, `log-search-references`, `log-intermediate-responses` are set to `false` by default and should remain so.

Here is an example to enable the Debug ACI Logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Debug ACI Logger" \
  --set enabled:true
```

Using this debug tracer is often more efficient by limiting the output using request and result criteria to match specific types of operations. An example result criteria for operations that fail due to insufficient access rights can be added to the logger as follows:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Debug ACI Logger" \
  --set "result-criteria:Insufficient Access Rights"
```

Once the logger is enabled, all matching operations begin writing ACI evaluation traces to the log file. The amount of information is quite large for each evaluation that is done. However, this information is useful if there is an ACI issue that is difficult to resolve. Most operations result in multiple "ACI DEBUG" traces in the log, since it usually requires multiple ACI rights to perform an operation, each of which requires a separate evaluation. In particular, you can expect a lot of debug tracing when dealing with ACIs for controls, extended operations, and proxied authorization.

The ACI DEBUG traces contain the following pieces of information:

- **Operation**. Specifies a dump of the operation object that you can use to correlate to the original request operation.
- **ACI Container**. Specifies the context of the ACI evaluation being performed.

  Client Entry. Specifies an LDIF dump of the client request access.
  Resource Entry. Specifies an LDIF dump of the target resource.
  isProxiedAuth. Specifies if the client is attempting to proxy as another user.
  Original Auth. Specifies the original client DN if authorization is currently via the proxy.
  Rights. Specifies a list of the ACI rights being requested on the resource entry.
  Control. Specifies the OIDs when evaluating ACIs for a control.
  ExtOp. Specifies the OIDs when evaluating ACIs for an extended operation.
- **ACI Canidates**. Specifies a list of all the ACIs known to this operation, sorted by origin.
- **Applicable ACIs**. Specifies a list of ACIs relevant to the current evaluation. These ACIs are separated by type into "Denies" and "Allows".
- **Deny ACI Evaluations**. Specifies the results of evaluating each "deny" ACI. If any of these evaluate to TRUE, then the operation will be denied.
- **Allow ACI Evaluations**. Specifies the results of evaluating each "allow" ACI. At least one of these must evaluate to "TRUE" or the operation will be denied.

For users with the `bypass-acl` privilege, the Debug ACI Logger will not provide any ACI debug tracing since evaluations are not done for those operations. However, you will see the following trace if you have ACI debugging enabled (`debug-aci-enabled` is set to TRUE) for those operations:

```
Bypassing ACL Evaluation for Operation
```

To avoid unnecessary tracing of these operations, the "Debug ACI Logger" uses a "Client Connection Criteria" called "Clients subject to Access Control" that excludes requests from users with the `bypass-acl` privilege. It is recommended that you create and use your own criteria which specifically targets the clients that you are trying to debug in order to make analyzing the tracing output easier.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Restricted Clients" \
  --type simple \
  --set none-included-user-privilege:bypass-acl
```

Problems with the Administrative Console

If a problem arises when trying to use the Administrative Console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with the Directory Proxy Server instance using the Administrative Console, then the access and error logs for that Directory Server instance might provide additional information about the underlying problem.

Problems with the Administrative Console: JVM Memory Issues

Console runs out of memory (PermGen). An inadequate PermSize setting in the server, while hosting web applications like the Administrative Console may result in errors like this in the error log:

```
[02/Mar/2016:07:50:27.017 -0600] threadID=2 category=UTIL
    severity=SEVERE_ERROR msgID=-1 msg="The server experienced an unexpected
    error. Please report this problem and include this log file.
    OutOfMemoryError: PermGen space
    ()\ncom.unboundid.directory.server.core.DirectoryServer.uncaughtException
    (DirectoryServer.java:15783)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1057)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.Thread.dispatchUncaughtException
    (Thread.java:1986)\nBuild revision: 22496\n"
```

This is only relevant for servers running Java 7.

Problems with the HTTP Connection Handler

When problems with the HTTP Connection Handler occur, first look at the HTTP connection handler log to diagnose the issue. The following section shows HTTP log examples when various errors occur.

- **Failed Request Due to a Non-Existent Resource**. The server receives a status code 404, which indicates the server could not match the URI.

  ```
  [15/Mar/2012:17:39:39 -0500] RESULT requestID=0 from="10.2.1.113:52958"
  method="GET" url="https://10.2.1.113:443/Aleph/Users/uid=user.1,ou=people,
  dc=example,dc=com" requestHeader="Host: x2270-11.example.lab"
  requestHeader="Accept: */*" requestHeader="User-Agent: curl/7.21.6
  (i386-pc-centos2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22
  libssh2/1.2.7" authorizationType="Basic" statusCode=404 etime=81.484
  ```

```
responseContentLength=103 responseHeader="Access-Control-Allow-
Credentials:true"
responseContentType="application/json"
```

- **Failed Request due to a Malformed Request Body**. The server receives a status code 400, which indicates that the request had a malformed syntax in its request body.

```
[15/Mar/2012:17:47:23-0500] RESULT requestID=10 from="10.2.1.113:55284"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
x2270-11.example.lab" requestHeader="Expect: 100-continue"
requestHeader="Accept: */*" requestHeader="Content-Type: application/json"
requestHeader="User-Agent: curl/ 7.21.6 (i386-pc-centos2.10) libcurl/7.21.6
OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
 authorizationType="Basic"
requestContentType="application/json" requestContentLength=5564
 statusCode=400
etime=15.272 responseContentLength=133 responseContentType="application/
json"
```

- **Failed Request due to an unsupported HTTP method**. The server receives a status code 405, which indicates that the specified method (e.g., "PATCH") in the request line is not allowed for the resource identified in the URI.

```
[15/Mar/2012:17:48:59-0500] RESULT requestID=11 from="10.2.1.113:55763"
method="PATCH" url="https://10.2.1.113:443/Aleph/Users"
 requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="Content-
Type:
application/json" requestHeader="User-Agent: curl/7.21.6 (i386-pc-
centos2.10)
libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
authorization-Type="Basic" requestContentType="application/json"
 statusCode=405
etime=6.807 responseContentLength=0 responseHeader="Allow: POST, GET,
 OPTIONS, HEAD"
```

- **Failed Request due to an Unsupported Media Type**. The server receives a status code 415, which indicates that the request entity is in a format that is not supported by the requested resource.

```
[15/Mar/2012:17:44:45-0500] RESULT requestID=4 from="10.2.1.113:54493"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="Content-
Type:
application/atom+xml" requestHeader="User-Agent: curl/7.21.6 (i386-pc-
centos2.10)
libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
authorizationType="Basic" requestContentType="application/atom+xml"
requestContentLength=3 statusCode=415 etime=6.222
 responseContentLength=1402
responseHeader="Cache-Control: must-revalidate,no-cache,no-store"
responseContentType="text/html;charset=ISO-8859-1"
```

- **Failed Request due to an Authentication Error**. The server receives a status code 401, which indicates that the request requires user authentication.

```
[15/Mar/2012:17:46:06-0500] RESULT requestID=8 from="10.2.1.113:54899"
method="GET" url="https://10.2.1.113:443/Aleph/Schemas"
 requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="User-
Agent:
curl/7.21.6 (i386-pc-centos2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5
libidn/1.22 libssh2/ 1.2.7" authorizationType="Basic" statusCode=401
etime=2.751 responseContentLength=63 responseHeader="WWW-Authenticate:
 Basic
realm=SCIM" responseHeader="Access-Control-Allow-Credentials: true"
responseContentType="application/json"
```

Virtual Process Size on RHEL6 Linux is Much Larger than the Heap

Red Hat Linux introduced a change in glib 2.11 that creates larger per-thread address spaces aligned at 64MB. This change results in a virtual process size much larger than those seen in previous versions of `glibc`. This is not considered a bug by RedHat, as noted in *https://bugzilla.redhat.com/show_bug.cgi?id=640286*, and does not affect the physical memory needed by the server process. To see the version of `glibc` on your system, use the command `yum info glibc`.

Providing Information for Support Cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the `collect-support-data` tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

Chapter 26

Command-Line Tools

Topics:

The PingDirectory Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

This chapter presents the following topics:

Using the Help Option

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. You can view detailed argument options and examples by typing --help with the command.

```
bin/dsconfig --help
```

For those utilities that support additional subcommands (for example, `dsconfig`), you can get a list of the subcommands by typing --help-subcommands.

```
bin/dsconfig --help-subcommands
```

You can also get more detailed subcommand information by typing --help with the specific subcommand.

```
bin/dsconfig list-log-publishers --help
```

Note: For detailed information and examples of the command-line tools, see the *Ping Identity Directory Server Command-Line Tool Reference*.

Available Command-Line Utilities

The Directory Server provides the following command-line utilities, which can be run directly in interactive, non-interactive, or script mode.

Table 70: Command-Line Utilities

Command-Line Tools	Description
audit-data-security	Performs an internal task that examines all or a subset of entries in the server, writing a series of reports on potential risks with the data. Reports are written to the output directory organized by backend name and audit items.
authrate	Perform repeated authentications against the Directory Server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more directory server backends. This utility also supports the use of a properties file to pass predefined command-line arguments. See *Managing the tools.properties File* for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a ZIP archive that can be sent to a technical support representative.
config-diff	Generate a summary of the configuration changes in a local or remote server instance. The tool can be used to compare configuration settings when troubleshooting issues, or when verifying configuration settings on new servers.
create-rc-script	Create a Run Control (RC) script that may be used to start, stop, and restart the Directory Server on UNIX-based systems.
create-recurring-task	Create a recurring task to run on the server. Tasks can be created for backups, LDIF exports, a statically defined task, or a third-party task.
create-recurring-task-chain	Create a chain of recurring tasks to run on the server.

Command-Line Tools	Description
dbtest	Inspect the contents of Directory Server backends that store their information in Oracle® Berkeley DB Java Edition databases.
deliver-one-time-password	Submit a "deliver one-time password" extended request, OID 1.3.6.1.4.1.30221.2.6.24, to the server which results in a the generation of a one-time password which is delivered out-of-band to the specified user. This tool can be used to test the UNBOUNDID-DELIVERED-OTP SASL mechanism.
dsconfig	View and edit the Directory Server configuration.
dsjavaproperties	Configure the JVM arguments used to run the Directory Server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and *JAVA_HOME* environment variable.
dsreplication	Manage data replication between two or more Directory Server instances.
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Directory Server.
encode-password	Encode user passwords with a specified storage scheme or determine whether a given clear-text value matches a provided encoded password.
encryption-settings	Manage the server encryption settings database.
enter-lockdown-mode	Request that the Directory Server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege.
export-ldif	Export data from the Directory Server backend in LDIF form.
identify-references-to-missing-entries	Identify entries containing one or more attributes that reference entries that do not exist. This may require the ability to perform unindexed searches and/or the ability to use the simple paged results control.
identify-unique-attribute-conflicts	Identify unique attribute conflicts. The tool may identify values of one or more attributes that are supposed to exist only in a single entry but are found in multiple entries.
import-ldif	Import LDIF data into the Directory Server backend.
ldap-diff	Compare the contents of two LDAP servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform LDAP compare operations in the Directory Server.
ldapdelete	Perform LDAP delete operations in the Directory Server.
ldapmodify	Perform LDAP modify, add, delete, and modify DN operations in the Directory Server.
ldappasswordmodify	Perform LDAP password modify operations in the Directory Server.
ldapsearch	Perform LDAP search operations in the Directory Server.
ldif-diff	Compare the contents of two LDIF files, the output being an LDIF file needed to bring the source file in sync with the target.
ldifmodify	Apply a set of modify, add, and delete operations against data in an LDIF file.
ldifsearch	Perform search operations against data in an LDIF file.
leave-lockdown-mode	Request that the Directory Server leave lockdown mode and resume normal operation.

Command-Line Tools	Description
list-backends	List the backends and base DNs configured in the Directory Server.
make-ldif	Generate LDIF data based on a definition in a template file.
manage-account	Access and alter password policy state properties for user entries.
manage-extension	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the PingDirectory Server. Extension bundles are installed from a zip archive or file system directory. PingDirectory Server will be restarted if running to activate the extension(s).
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Directory Server.
migrate-ldap-schema	Migrate schema information from an existing LDAP server into a PingDirectory Server instance.
migrate-sun-ds-config	Update an instance of the PingDirectory Server to match the configuration of an existing Sun Java System 5.x, 6.x, or 7.x directory instance.
modrate	Perform repeated modifications against a Directory Server.
move-subtree	Move a subtree entries or a single entry from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
profile-viewer	View information in data files captured by the Directory Server profiler.
re-encode-entries	Initiate a task that causes a local DB backend to re-encode all or a specified subset of the entries that it contains. The tool does not alter the entries themselves but provides a useful mechanism for applying significant changes to the way that entries are stored in the backend (e.g., to apply encoding changes if a feature like data encryption or uncached attributes or entries is enabled).
rebuild-index	Rebuild index data within a backend based on the Berkeley DB Java Edition. Note that this tool uses different approaches to rebuilding indexes based on whether it is running in online mode (as a task) rather than in offline mode. Running in offline mode will often provide significantly better performance. Also note that running in online mode will prevent the server from using that index while the rebuild is in progress, so some searches may behave differently while a rebuild is active than when it is not.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Directory Server backend.
restore	Restore a backup of the Directory Server backend.
revert-update	Returns a server to the version before the last update was performed.
review-license	Review and/or indicate your acceptance of the product license.
scramble-ldif	Obscure the contents of a specified set of attributes in an LDIF file.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
searchrate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Directory Server process.
setup	Perform the initial setup for the Directory Server instance.
start-server	Start the Directory Server.

Command-Line Tools	Description
status	Display basic server information.
stop-server	Stop or restart the Directory Server.
subtree-accessibility	List or update the a set of subtree accessibility restrictions defined in the Directory Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
uninstall	Uninstall the Directory Server.
update	Update the Directory Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the `update` tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update is performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the `revert-update` tool for information on reverting an update.
validate-acis	Validates a set of access control definitions contained in an LDAP server (including Sun/Oracle DSEE instances) or an LDIF file to determine whether they are acceptable for use in the Directory Server. Note that the output generated by this tool will be in LDIF format, but each entry in the output will have exactly one ACI, so entries that have more than one ACI will appear multiple times in the output with different ACI values.
validate-file-signature	Validate the signature information in a signed text file, such as a signed log file or a signed LDIF export.
validate-ldif	Validate the contents of an LDIF file against the server schema.
verify-index	Verify that indexes in a backend using the Oracle Berkeley DB Java Edition are consistent with the entry data contained in the database.

Managing the tools.properties File

The PingDirectory Server supports the use of a tools properties file that simplifies command-line invocations by reading in a set of arguments for each tool from a text file. Each property is in the form of name/value pairs that define predetermined values for a tool’s arguments. Properties files are convenient when quickly testing the Directory Server in multiple environments.

The Directory Server supports two types of properties file: default properties files that can be applied to all command-line utilities or tool-specific properties file that can be specified using the --propertiesFilePath option. You can override all of the Directory Server's command-line utilities with a properties file using the `config/tools.properties` file.

Creating a Tools Properties File

You can create a properties file with a text editor by specifying each argument, or option, using standard Java properties file format (name=value). For example, you can create a simple properties file that define a set of LDAP connection parameters as follows:

```
hostname=server1.example.com
port=1389
```

```
bindDN=cn=Directory\ Manager
bindPassword=secret
baseDN=dc=example,dc=com
```

Next, you can specify the location of the file using the --propertiesFilePath /path/to/ File option with the command-line tool. For example, if you save the previous properties file as `bin/mytool.properties`, you can specify the path to the properties file with `ldapsearch` as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/mytools.properties "(objectclass=*)"
```

Properties files do not allow quotation marks of any kind around values. Any spaces or special characters should be escaped. For example,

```
bindDN=cn=QA\ Managers,ou=groups,dc=example,dc=com
```

The following is not allowed as it contains quotation marks:

```
bindDN=cn="QA Managers,ou=groups,dc=example,dc=com"
```

Tool-Specific Properties

The Directory Server also supports properties for specific tool options using the format: `tool.option=value`. Tool-specific options have precedence over general options. For example, the following properties file uses `ldapsearch.port=2389` for `ldapsearch` requests by the client. All other tools that use the properties file uses `port=1389`.

```
hostname=server1.example.com
port=1389
ldapsearch.port=2389
bindDN=cn=Directory\ Manager
```

Another example using the `dsconfig` configuration tool is as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
dsconfig.bindPasswordFile=/ds/config/password
```

Note: The `.bindPasswordFile` property requires an absolute path. If you were to specify `~/ds/config/password`, where ~ refers to the home directory, the server does not expand the ~ value when read from the properties file.

Specifying Default Properties Files

The Directory Server provides a default properties files that apply to all command-line utilities used in client requests. A default properties file, `tools.properties`, is located in the `<server-root>/config` directory.

If you place a custom properties file that has a different filename as `tools.properties` in this default location, you need to specify the path using the --propertiesFilePath option. If you make changes to the `tools.properties` file, you do not need the --propertiesFilePath option. See the examples in the next section.

Evaluation Order Summary

The Directory Server uses the following evaluation ordering to determine options for a given command-line utility:

- All options used with a utility on the command line takes precedence over any options in any properties file.
- If the --propertiesFilePath option is used with no other options, the Directory Server takes its options from the specified properties file.
- If no options are used on the command line including the --propertiesFilePath option (and --noPropertiesFile), the Directory Server searches for the `tools.properties` file at `<server-root>`
- If no default properties file is found and a required option is missing, the tool generates an error.

- Tool-specific properties (for example, `ldapsearch.port=3389`) have precedence over general properties (for example, `port=1389`).

Evaluation Order Example

Given the following properties file that is saved as `<server-root>/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

The Directory Server locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (port and baseDN). The command uses the `bindDN` and `bindPassword` arguments specified in the properties file.

   ```
   $ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
     --propertiesFilePath bin/tools.properties “(objectclass=*)”
   ```
2. Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the Directory Server uses the specified properties file as follows:

   ```
   $ bin/ldapsearch --propertiesFilePath bin/tools.properties \
     “(objectclass=*)”
   ```
3. If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, the Directory Server attempts to locate any default `tools.properties` file in the following location:

   ```
   <server-root>/config/tools.properties
   ```

   Assume that you move your tools.properties file from `<server-root>/bin` to the `<server-root>/config` directory. You can then run your tools as follows:

   ```
   $ bin/ldapsearch "(objectclass=*)"
   ```

   The Directory Server can be configured so that it does not search for a properties file by using the --noPropertiesFile option. This options tells the Directory Server to use only those options specified on the command line. The --propertiesFilePath and --noPropertiesFile options are mutually exclusive and cannot be used together.
4. If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Running Task-based Utilities

The Directory Server has a Tasks subsystem that allows you to schedule basic operations, such as backup, restore, bin/start-server, bin/start-server and others. All task-based utilities require the --task option that explicitly indicates the utility is intended to run as a task rather than in offline mode. The following table shows the arguments that can be used for task-based operations:

Table 71: Task-based Utilities

Option	Description
--task	Indicates that the tool is invoked as a task. The --task argument is required. If a tool is invoked as a task without this --task argument, then a warning message will be displayed stating that it must be used. If the --task argument is provided but the tool was not given the appropriate set of authentication arguments to the

Option	Description
	server, then an error message will be displayed and the tool will exit with an error.
--start	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation starts when scheduled as a server task. A value of '0' causes the task to be scheduled for immediate execution. When this option is used, the operation is scheduled to start at the specified time, after which this utility will exit immediately.
--dependency	Specifies the ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution. This option can be used multiple times in a single command.
--failedDependencyAction	Specifies the action this task will take should one of its dependent tasks fail. The value must be one of the following: `PROCESS`, `CANCEL`, `DISABLE`. If not specified, the default value is `CANCEL`. This option can be used multiple times in a single command.
--completionNotify	Specifies the email address of a recipient to be notified when the task completes. This option can be used multiple times in a single command.
--errorNotify	Specifies the email address of a recipient to be notified if an error occurs when this task executes. This option can be used multiple times in a single command.

PingDirectoryProxy™

Release 7.0.1

Server Administration Guide

Notice

PingDirectory™ Product Documentation

Trademarks

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Contents

Chapter 1

Introduction

Topics:

PingDirectoryProxy™ Server is a fast and scalable LDAPv3 gateway for the PingDirectory® Server. The Directory Proxy Server architecture can be configured to control how client requests are routed to backend servers.

This chapter provides an overview of the Directory Proxy Server features and components. It contains the following sections:

Overview of the PingDirectoryProxy Server Features

The PingDirectoryProxy Server is a fast, scalable, and easy-to-use LDAP proxy server that provides high availability and additional security for the PingDirectory Server, while remaining largely invisible to client applications. From a client perspective, request processing is the same, whether communicating with the Directory Server directly or going through the Directory Proxy Server.

The PingDirectoryProxy Server provides the following set of features:

- **High availability**. The Directory Proxy Server allows you to transparently fail over between servers if a problem occurs, as well as ensuring that the workload is balanced across the topology. If a client does not support following referrals, the Directory Proxy Server can follow them on the client’s behalf.
- **Data mapping and transformation**. The Directory Proxy Server can do DN mapping and attribute mapping to allow clients to interact with the server using older names for directory content. It allows clients to continue working when they would not be able to work directly with the Directory Server, either because of changes that have occurred at the data layer or to inherent design limitations in the clients.
- **Horizontal scalability and performance**. Reads can be horizontally scaled using load balancing. In large data centers, if the data set is too large to be cached or to provide horizontal scalability for writes, the Directory Proxy Server can automatically split the data across multiple systems. This feature allows the Directory Proxy Server to improve scalability and performance of the Directory Server environment.
- **Load balancing and failover**. You can spread the workload across multiple proxies in a large data center using load-balancing algorithms. Load balancing is also useful when a server becomes degraded or non-responsive, because client process requesting is directed to a different server.
- **Security and access control**. The Directory Proxy Server can add additional firewall capabilities, as well as constraints and filtering to help protect the Directory Server from attacks. You can use a Directory Proxy Server in a DMZ as opposed to allowing clients to directly access the Directory Proxy Server in the internal network or providing the data in the DMZ. It can help provide secure access to the data and you can define what actions clients are allowed to do. For example, you can prevent clients from making modifications to data when connected via a VPN no matter what their identity or permissions.
- **Tracking of operations across the environment**. In the past, administrators have commonly complained that when they see a request in the access log, they have no idea where it came from and cannot track it back to a particular client. The Directory Proxy Server contains controls that allow administrators to track requests back to the client that issued them. Whenever the Directory Proxy Server forwards a request to the Directory Server, it includes a control in the request so that the Directory Server's access log has the IP address of the client, address and connection ID of the Directory Server. In the response back to the client, it similarly includes information about the Directory Server that processed the request, such as the connection ID and operation ID. This feature makes it easier for administrators to keep track of what is going on in their environment.
- **Monitoring and management tools**. Because the Directory Proxy Server uses many of the components of the PingDirectory Server, it can leverage them to provide protocol support, logging, management tools for configuration and monitoring, schema, and so on. You can use the Data Metrics Server, the `dsconfig` tool and the Administrative Console to manage the Directory Proxy Server.

Overview of the Directory Proxy Server Components and Terminology

The Directory Proxy Server consists of the following components and functionality that provide the proxy capabilities:

Locations
LDAP External Servers
LDAP Health Checks
Load-Balancing Algorithms
Data Transformations
Request Processors
Server Affinity Providers

Subtree Views
Connection Pools
Client Connection Policies
Entry Balancing

This section describes each component in more detail.

About Locations

Locations define a group of servers with similar response time characteristics. Each location consists of a name and an ordered list of preferred failover locations. The Directory Proxy Server and each of the backend LDAP external servers can be assigned locations. These locations can be taken into account when deciding how to route requests, so that the server prefers to forward requests to Directory Server in the same data center over those in remote locations. As a rule of thumb, if you have multiple data centers then you should have a separate location for each one. In most environments, all Directory Proxy Server instances should have the same configuration except for the attribute that specifies the location of the Directory Proxy Server itself.

For example, a deployment consists of three data centers, one in New York, another in Chicago, and another in Los Angeles. In the New York data center, applications which reside in this data center prefer communicating with directories in this data center. If none of the servers are available, it prefers to failover to the data center in Chicago rather than the data center in Los Angeles. So the New York location contains an ordered list in which the Chicago location is preferred over the Los Angeles data center for failover.

For information about configuring locations, see *Configuring Locations*.

About LDAP External Servers

You can configure information about the directory server instances accessed by the PingDirectoryProxy Server. This configuration information includes the following:

Server connection information, such as IP address, port, and security layer
Location
Authentication information
Methods for authenticating and authorizing clients
Server-specific health checks
Types of operations allowed. For example, some LDAP external servers may allow only reads and others allow reads and writes, so the Directory Proxy Server can recognize this and accommodate it.

The PingDirectoryProxy Server allows you to configure different types of LDAP external servers. The default configuration for each type is tuned to be the best possible configuration for each.

For information about configuring LDAP external servers, see *Configuring LDAP External Servers*.

About LDAP Health Checks

The LDAP health check component provides information about the availability of LDAP external servers. The health check result includes a server state, which can be one of the following:

- **Available**. Completely accessible for use.
- **Degraded**. The server may be used if necessary, but has a condition which may make it less desirable than other servers (for example, it is slow to respond or has fallen behind in replication).
- **Unavailable**. Completely unsuitable for use (for example, the server is offline or is missing critical data).

Health check results also include a numeric score, which has a value between 1 and 10, that can help rank servers with the same state. For example, if two servers are available and one has a score of 8 and the other a score of 7, the Directory Proxy Server can be configured to prefer the server with the higher score.

The Directory Proxy Server periodically invokes health checks to monitor each LDAP external server, and may also initiate health checks in response to failed operations. It checks the health of the LDAP external servers at intervals configured in the LDAP server's `health-check-frequency` property. However, the Directory Proxy Server has safeguards in place to ensure that only one health check is in progress at any time against a backend server to avoid affecting its ability to process other requests.

The results of health checks performed by the Directory Proxy Server are made available to the load-balancing algorithms so that they may be taken into account when determining where to send requests. The Directory Proxy Server will attempt to use servers with a state of available before trying servers with a state of degraded. It will never attempt to use servers with a state of unavailable. Some load-balancing algorithms may also take the health check score into account, such as the health-weighted load-balancing algorithm, which prefers servers with higher scores over those with lower scores. You configure the algorithms that work best for you environment.

In some cases, an LDAP health check may define different sets of criteria for promoting and demoting the state of a server. So, a degraded server may need to meet more stringent require- ments to be reclassified as available than it originally took to be considered degraded. For example, if response time is used in the process of determining the health of a server, then the Directory Proxy Server may have a faster response time threshold for transitioning a server from degraded back to available than the threshold used to consider it degraded in the first place. This threshold difference can help avoid cases in which a server repeatedly transitions between the two states because it is operating near the threshold.

For example, the health check used to measure search response time is configured to mark any server to be marked degraded when the search response time is greater than 1 second. You can then configure that the response time must be less than 500 ms before the server is made available again, so that the Directory Proxy Server does not flip back and forth between available and degraded.

PingDirectoryProxy Server provides the following health checks:

- **Measure the response time for searches and examine the entry contents**. For example, the health check might retrieve a monitoring entry from a server and base the health check result on whether the entry was returned, how long it took to be returned, and whether the value of the returned entry matches what was expected.
- **Monitor the replication backlog**. If a server falls too far behind in replication, then the Directory Proxy Server can stop sending requests to it. A server is classified as degraded or unavailable if the threshold is reached for the number of missing changes, the age of the oldest missing change, or both.
- **Consume Directory Server administrative alerts**. If the Directory Server indicates there is a problem, for example an index that must be rebuilt, then it will flag itself as degraded or unavailable. When the Directory Proxy Server detects this, it will stop sending requests to the server. The Directory Proxy Server detects administrative alerts as soon as they are issued by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the Directory Server. When the Directory Proxy Server is notified by the Directory Server of a new alert, it immediately retrieves the base `cn=monitor` entry of the Directory Server. If this entry has a value for the `unavailable-alert-type` attribute, then the Directory Proxy Server will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the Directory Proxy Server will consider it degraded. Clients of the Directory Proxy Server can use a similar mechanism to detect and react when a Directory Proxy Server flags itself as degraded or unavailable.
- **Monitor the busyness of the server**. If a server becomes too busy, then it may be marked degraded or unavailable so that less heavily-loaded servers may be preferred.

For information about configuring health checks, see *Configuring Server Health Checks*. To associate a health check with an LDAP external server and set the health check frequency, you must configure the `health-check` and `health-check-frequency` properties of the LDAP external server. See “To Configure an External Server Using dsconfig” for information about configuring the properties of the external server.

About Load-Balancing Algorithms

Load-balancing algorithms are used to determine which server in a set of similar servers should be used to process a client request. The algorithm can take the following criteria into account:

- **Consider the location of the server**. Servers in the same location as the Directory Proxy Server can be preferred over those in alternate locations.
- **Consider the health of the server**. Servers that are available are preferred over those that are degraded. In some cases, the health check score may also be used to further differentiate between servers with the same health check state.

- **Route requests consistently**. Requests from a single client may be consistently routed to the same directory server instance to avoid problems such as propagation delay from replication.
- **Retry the operation in an alternate server if the request fails or the operation times out**. You can control if the retry is allowed and, if so, how many times to retry and the time out interval.

The PingDirectoryProxy Server provides the following load-balancing algorithms:

- **Fewest operations**. Requests are forwarded to the backend server with the fewest operations currently in progress.
- **Single server**. Requests are always sent to the same server and will not attempt to fail over to another server if the target server is unavailable.
- **Weighted**. Administrators explicitly assign numeric weights to individual servers or sets of servers to control how likely they are to be selected for processing requests relative to other servers.
- **Health-based weighting**. Uses the health check score to assign weights to each of the servers, so that a server with a higher score gets a higher percentage of the traffic than a server with a lower score. The proportion of traffic received is the difference between their health check scores.
- **Failover**. Requests are always sent to a given server first. If that server fails, then the request is sent to another specified server, and so on through an ordered failover server list.

For information about configuring load balancing, see *Configuring Load Balancing*.

About Proxy Transformations

Proxy transformations are used to rewrite requests and responses as they pass through the Directory Proxy Server. Proxy data transformations are helpful for clients that use an old schema or that contain a hard-coded schema.

Proxy transformations can provide DN and attribute mapping altering both requests to the server as well as responses from the server. For example, a client sends a request to `o=example.com` even though the directory server handling the request uses `dc=example,dc=com`. The Directory Proxy Server can transparently remap the request so that the server can process it, and map it back to the original DN of the client request when the value is returned. Or if a client tries to use the attribute `userID`, the Directory Proxy Server can map it to `uid` before sending the request on to the backend LDAP server. The Directory Proxy Server then remaps the response to `userID` when the value is returned.

The Directory Proxy Server also includes a proxy transformation that can be used to suppress a specified attribute, so that it will never be returned to clients. It can also cause the server to reject requests which target that particular attribute. Another proxy transformation can be used to prevent entries that match a given search filter from being returned to clients.

For information about configuring proxy transformations, see *Configuring Proxy Transformations* on page 70.

About Request Processors

A request processor encapsulates the logic for handling an operation, ensuring that a given operation is handled appropriately. The request processor can either process the operation directly, forward the request to another server, or hand off the request to another request processor.

PingDirectoryProxy Server provides the following types of request processor:

- **Proxying request processors**, which forward operations received by the Directory Proxy Server to other LDAP external servers.
- **Entry-balancing request processors**, which split data across multiple servers. They determine which set of servers are used to process a given operation. They then hand off operations to proxying request processors so that requests can be forwarded to one of the servers in the set.
- **Failover request processors**, which perform ordered failover between other types of request processors, sometimes with different behavior for different types of operations.

Directory Proxy Server request processors can be used to forward certain controls, including the batch transaction control and the LDAP join control. The batch transaction control must target a single Berkley DB backend. For more information about the controls, refer to the LDAP SDK for Java documentation.

For information about configuring request processors, see *Configuring Request Processors* on page 72.

About Server Affinity Providers

The server affinity provider can be used to establish an affinity to a particular backend server for certain operations. You can configure one of three types of provider:

- **Client connection Server Affinity**, so that requests from the same client connection may consistently be routed to the same backend server.
- **Client IP address Server Affinity**, so that all requests coming from the same client system will be consistently routed to the same backend server.
- **Bind DN Server Affinity**, so that all requests from the same user will be consistently routed to the same backend server.

For information about configuring server affinity, see *Configuring Server Affinity*.

About Subtree Views

A subtree view can be used to make a portion of the DIT available to a client by associating a request processor with a base DN. Subtree views allow you to route operations concerning one set of data to a particular set of data sources, and operations concerning another set of data to another set of data sources. Multiple subtree views may be involved in processing a request, such as for searches that have a scope that is larger than the subtree view.

The subtree view includes a single base DN used to identify the portion of the DIT. They may have hierarchical relationships, for example one subtree view could be configured for `dc=example,dc=com` and another for `ou=People,dc=example,dc=com`.

For information about configuring a subtree view, see *Configuring Subtree Views*.

About the Connection Pools

Based on the type of backend server that you are using, the PingDirectoryProxy Server maintains either one or two connection pools to the backend server. It maintains either one pool for all types of operations or two separate pools for processing bind and non-bind operations from clients. When the Directory Proxy Server establishes connections, it authenticates them using whatever authentication mechanism is defined in the configuration of the external server. These connections will be re-used for all types of operations to be forwarded to the backend server. The bind DN and password are configured in the Directory Proxy Server.

Whenever a client sends a bind request to the Directory Proxy Server, the server looks at the type of bind request that was sent. If it is a SASL bind request, then the authentication is processed by the Directory Proxy Server itself and it will not be forwarded to the backend server. However, the Directory Proxy Server may use information contained in the backend server as needed. If the bind request is a simple bind request and the bind DN is within the scope of data supplied by the backend server, then the Directory Proxy Server will forward the client request to the backend server so that it will use the credentials provided by the client.

Regardless of the authentication method that the client uses, the Directory Proxy Server will remember the identity of the client after the authentication is complete and for any subsequent requests sent by that client, it will use the configured authorization method to identify the client to the backend server. Even though the operation is forwarded over a connection that is authenticated as a user defined in the Directory Proxy Server configuration, the request is processed by the backend server under the authority of the end client.

About Client Connection Policies

Client connection policies define the general behavior the server exhibits when communicating with a set of clients. Each policy consists of the following:

- A **set of connection criteria** that define which client is associated with the policy based on information the server has about the client, including client address, protocol used, secure communication mechanism, location of the client's entry in the Directory Server and the contents of the client's entry. These criteria are the same as those used for filtered logging. For example, different client connection policies could be established for different classes of users, such as root and non-root users.

- A **set of constraints** on the type of operations a client may request. You can specify whether a particular type of operation is allowed for clients. For some operation types, such as extended operations, you can allow only a particular subset of an operation type, such as a particular extended operation.
- A **set of subtree views** that define information about the parts of the DIT the client may access.

When a client connection is established, only one client connection policy is applied. If the criteria for several policies match the same client connection, the evaluation order index is used as a tiebreaker. If no policy matches, the client connection is terminated. If the client binds, changing its identity, or uses StartTLS to convert from an insecure connection to a secure connection, then the connection may be evaluated again to determine if it matches the same or a different client connection policy. The connection can also be terminated if it no longer matches any policy.

For information about configuring a client connection policy, see *Configuring Client Connection Policies* on page 77.

About Entry Balancing

Entry balancing allows you to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance. Entry balancing can take advantage of a global index, an in-memory cache used to quickly determine which set or sets of servers should be used to process a request based on the entry DNs and/or the attribute values used in the request.

For information about configuring entry balancing, see *Deploying an Entry-Balancing Proxy Configuration* on page 148.

Server Component Architecture

This section provides an overview of the process flow between the Directory Proxy Server components, for both a simple proxy deployment and an entry-balancing deployment.

Architecture of a Simple Directory Proxy Server Deployment

In a simple Directory Proxy Server deployment, a client request is first processed by a client connection policy as illustrated in Figure 1, "Process Flow for Directory Proxy Server".

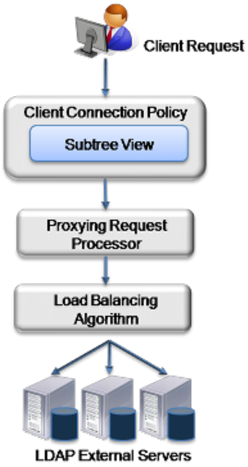


Figure 1: Process Flow for Directory Proxy Server

The client connection policy contains a subtree view, which defines the portion of the DIT available to clients. Once the Directory Proxy Server determines that the DIT is available, it passes the request to the request processor, which defines the logic for processing the request. The request processor then passes the request to a load-balancing algorithm, which determines the server in a set of servers responsible for handling the request. Finally, the request is passed to the LDAP external server. The LDAP external server contains properties that define the server's location in a topology and the health checks used to determine if the server is functioning properly. This information may be used by the load-balancing algorithm in the course of determining how to route requests.

Architecture of an Entry-Balancing Directory Proxy Server Deployment

Figure 2, “Process Flow for Entry-Balancing Directory Proxy Server” describes how a client request is treated in an entry-balancing deployment.

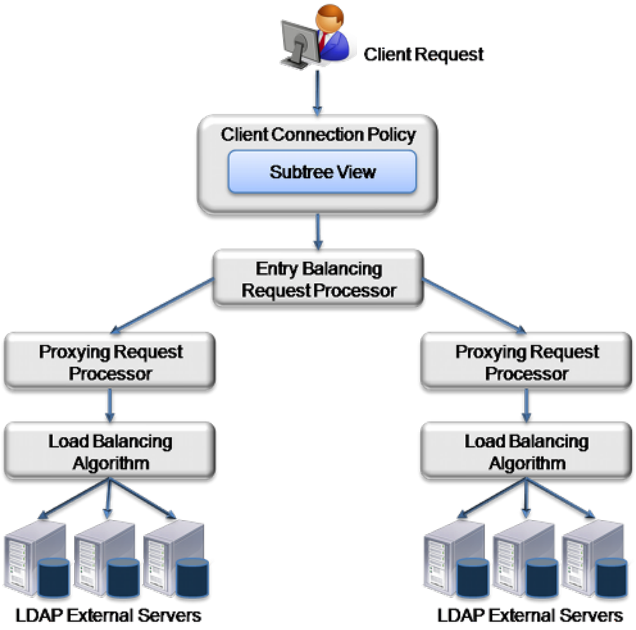


Figure 2: Process Flow for Entry-Balancing Directory Proxy Server

Entry balancing is typically used when the data set it too large to fully cache on a single server or when the write performance requirements of an environment are higher than can be achieved with a single replicated set of servers. In such cases, the data may be split across multiple sets of servers, increasing the memory available for caching and the overall write performance in proportion to the number of server sets.

As with a simple proxy deployment, the client request is first processed by the client connection policy, which determines how the Directory Proxy Server communicates with a set of clients. It contains a subtree view that represents the base DN for the entire deployment. The data set splits beneath this base DN.

The request is then passed to the entry-balancing request processor. The entry-balancing request processor contains a global attribute index property, which helps the request processor determine which server set contains the entry and how to properly route the request. It also contains a placement algorithm, which helps it select the server set in which to place new entries created by add requests.

Beneath the entry-balancing request processor are multiple proxying request processors that handle multiple unique sets of data. These request processors pass the request to a load-balancing algorithm, which determines which LDAP external server should handle the request. As with a simple proxy deployment, this LDAP external server contains properties that define the server’s location and the health checks used to determine if the server is functioning properly.

For configuration information, see *Configuring an Entry-Balancing Directory Proxy Server Deployment*. For information about entry-balancing replication, see *Overview of Replication in an Entry-Balancing Environment* on page 170.

Directory Proxy Server Configuration Overview

The configuration of the Directory Proxy Server involves the following steps:

- **Configuring the locations for your deployment**. A location is a collection of servers that share access and latency characteristics. For example, your deployment might include two data centers, one in the east and one in the west. These data centers would be configured as two locations in the Directory Proxy Server. Each location is associated with a name and an ordered list of failover locations, which could be used if none of the servers in the preferred location are available.

- **Configuring the Directory Proxy Server location**. You need to update the configuration to specify the location of the Directory Proxy Server instance.
- **Configuring health checks for the LDAP external servers**. You can configure at what point the Directory Proxy Server considers an LDAP external server to be available, of degraded availability, or unavailable. Each health check can be configured to be used automatically for all LDAP external servers or for a specified set of servers.
- **Configuring the LDAP external servers**. During this step, you define each of the external directory servers, including the server type. You can configure Ping Identity Directory Servers, Sun Java System Directory Servers, or generic LDAP servers. You also assign the server-specific health checks configured in the previous step.
- **Configuring the load-balancing algorithm**. You configure the load-balancing algorithm used by the Directory Proxy Server to determine which server in a set of similar servers should be used to process a client request. The Directory Proxy Server provides default algorithms. It also steps you through the creation of new algorithms by using an existing algorithm as a template or by creating one from scratch.
- **Configuring the proxying request processor**. In this step, you configure proxying request processors that forward operations received by the Directory Proxy Server to other LDAP external servers.
- **Configuring subtree views**. A subtree view defines the portion of the DIT available to a client. Each subtree view can be associated with a load-balancing algorithm to help distribute the work load.
- **Configuring the client connection policy**. You configure policies to classify how different client connections are managed by the Directory Proxy Server. The client connection policy can be used to control the types of operations that a client may perform and the portion of the DIT that the client can access. Restrictions configured in a client connection policy will take precedence over any capabilities granted by access control or privileges.

Chapter 2

Installing the Directory Proxy Server

Topics:

This section describes how to install PingDirectoryProxy Server. It includes pre-installation requirements and considerations.

It includes the following sections:

Before You Begin

The following sections describe requirements and considerations you should make before installing the software and configuring the PingDirectoryProxy Server objects.

> ⚠ **Important:**
>
> **Each Server Deployment Requires an Execution of Setup - Duplicating a Server-root is not Supported**. The installation of the server does not write or require any data outside of the server-root directory. After executing `setup`, copying the server-root to another location or system, in order to *duplicate* the installation, is not a supported method of deployment. The server-root can be moved to another host or disk location if a host or file system change is needed.

It is also highly recommended that a Network Time Protocol (NTP) system be in place so that multi-server environments are synchronized and timestamps are accurate.

Defining a Naming Strategy for Server Locations

The various objects defined in the PingDirectoryProxy Server will be specific to a particular location. Location names are used to define a grouping of PingDirectoryProxy Server products based on physical proximity. For example, a location is most often associated with a single datacenter location. During the installation, assign a location to each server for optimal inter-server behavior. The location assigned to a server within Global Configuration can be referenced by components within the server as well as processes external to the server to satisfy "local" versus "remote" decisions used in replication, load balancing, and failover.

Software Requirements: Java

For optimized performance, the PingDirectoryProxy Server requires Java for 64-bit architectures. You can view the minimum required Java version on your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

Even if your system already has Java installed, you may want to create a separate Java installation for use by the PingDirectoryProxy Server to ensure that updates to the system-wide Java installation do not inadvertently impact the Directory Proxy Server. This setup requires that the JDK, rather than the JRE, for the 64-bit version, be downloaded.

To Install Java (Oracle/Sun)

1. Open a browser and navigate to the Oracle download site.
2. Download the latest version Java JDK. Click the JDK Download button corresponding to the latest Java update.
3. On the Java JDK page, click the Accept Licence Agreement button, then download the version based on your operating system.

Preparing the Operating System

You should make the following changes to your operating system depending on the production environments on which the PingDirectoryProxy Server will run.

Configuring the File Descriptor Limits

The PingDirectoryProxy Server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. If needed, increase the file descriptor limit on the operating system.

If the operating system relies on `systemd`, refer to the Linux operating system documentation for instructions on setting the file descriptor limit.

To Set the File Descriptor Limit (Linux)

The Directory Proxy Server allows for an unlimited number of connections by default but is restricted by the file descriptor limit on the operating system. Many Linux distributions have a default file descriptor limit of 1024 per process, which may be too low for the server if it needs to handle a large number of concurrent connections.

Once the operating system limit is set, the number of file descriptors that the server will use can be configured by either using a `NUM_FILE_DESCRIPTORS` environment variable, or by creating a `config/num-file-descriptors` file with a single line such as, `NUM_FILE_DESCRIPTORS=12345`. If these are not set, the default of 65535 is used. This is strictly optional if wanting to ensure that the server shuts down safely prior to reaching the file descriptor limit.

1. Display the current hard limit of your system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

   ```
   ulimit -aH
   ```
2. Edit the `/etc/sysctl.conf` file. If there is a line that sets the value of the `fs.file-max` property, make sure its value is set to at least 65535. If there is no line that sets a value for this property, add the following to the end of the file:

   ```
   fs.file-max = 65535
   ```
3. Edit the `/etc/security/limits.conf` file. If the file has lines that sets the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before “#End of file”). Also note that you should insert a tab, rather than spaces, between the columns.

   ```
   * soft nofile 65535
   * hard nofile 65535
   ```
4. Reboot your system, and then use the `ulimit` command to verify that the file descriptor limit is set to 65535.

   ```
   # ulimit -n
   ```

Note: For RedHat 7 or later, modify the `20-nproc.conf` file to set both the open files and max user processes limits:

```
/etc/security/limits.d/20-nproc.conf

 Add or edit the following lines if they do not already exist:

 *           soft    nproc     65536
 *           soft    nofile    65536
 *           hard    nproc     65536
 *           hard    nofile    65536
 root        soft    nproc     unlimited
```

Enabling the Server to Listen on Privileged Ports (Linux)

Linux systems have a mechanism called capabilities that is used to grant specific commands the ability to do things that are normally only allowed for a root account. It may be convenient to enable the server to listen on privileged ports while running as a non-root user.

The `setcap` command is used to assign capabilities to an application. The `cap_net_bind_service` capability enables a service to bind a socket to privileged ports (port numbers less than 1024). If Java is installed in `/ds/java` (and the Java command to run the server is `/ds/java/bin/java`), the Java binary can be granted the `cap_net_bind_service` capability with the following command:

```
$ sudo setcap cap_net_bind_service=+eip /ds/java/bin/java
```

The java binary needs an additional shared library (`libjli.so`) as part of the Java installation. More strict limitations are imposed on where the operating system will look for shared libraries to load for commands that have capabilities assigned. So it is also necessary to tell the operating system where to look for this library. This can be done by creating the file /etc/ld.so.conf.d/libjli.conf with the path to the directory that contains the libjli.so file. For example, if the Java installation is in /ds/java, the contents of that file should be:

```
/ds/java/lib/amd64/jli
```

Run the following command for the change to take effect:

```
$ sudo ldconfig -v
```

To Set the Filesystem Flushes

With the out-of-the-box settings on Linux systems running the ext3 filesystem, the data is only flushed to disk every five seconds. If the Directory Proxy Server is running on a Linux system using the ext3 filesystem, consider editing the mount options for that filesystem to include the following:

```
commit=1
```

This variable changes the flush frequency from five seconds to one second.

You should also set the flush frequency to the /etc/fstab file. Doing the change via the mount command alone will not survive across reboots.

Disable Filesystem Swapping

It is recommended that any performance tuning services like tuned be disabled. If performance tuning is required, vm.swappiness can be set by cloning the existing performance profile then adding vm.swappiness = 0 to the new profile's tuned.conf file. This file is located at /usr/lib/tuned/profile-name/tuned.conf. The updated profile is then selected by running tuned-adm profile customized_profile.

About Editing OS-Level Environment Variables

Certain environment variables can impact the Directory Proxy Server in unexpected ways. This is particularly true for environment variables that are used by the underlying operating system to control how it uses non-default libraries.

For this reason, the Directory Proxy Server explicitly overrides the values of key environment variables like *PATH*, *LD_LIBRARY_PATH*, and *LD_PRELOAD* to ensure that something set in the environments that are used to start the server does not inadvertently impact its behavior.

If there is a legitimate need to edit any of these environment variables, the values of those variables should be set by manually editing the set_environment_vars function of the lib/_script-util.sh script. You will need to stop (bin/stop-server) and re-start (bin/start-server) the server for the change to take effect.

Install sysstat and pstack (Red Hat)

For Red Hat® Linux systems, you should install a couple of packages, sysstat and pstack, that are disabled by default, but are useful for troubleshooting purposes in the event that a problem occurs. The troubleshooting tool collect-support-data uses the iostat, mpstat, and pstack utilities to collect monitoring, performance statistics, and stack trace information on the server's processes. For Red Hat systems, make sure that these packages are installed, for example:

```
$ sudo yum install sysstat gdb dstat -y
```

Install dstat (SUSE Linux)

The dstat utility is used by the collect-support-data tool and can be obtained from the OpenSuSE project website. The following example shows how to install the dstat utility on SuSE Enterprise Linux 11 SP2:

1. Login as Root.
2. Add the appropriate repository using the zypper tool.

3. Install the `dstat` utility.

   ```
   $ zypper install dstat
   ```

Omit vm.overcommit_memory

Administrators should be aware that an improperly configured value for the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file can cause the `setup` or `start-server` tool to fail.

For Linux systems, the `vm.overcommit_memory` property sets the kernel policy for memory allocations. The default value of 0 indicates that the kernel determines the amount of free memory to grant a `malloc` call from an application. If the property is set to a value other than zero, it could lead the operating system to grab too much memory, depriving memory for the `setup` or `start-server` tool.

We recommend omitting the property in the `/etc/sysctl.conf` file to ensure that enough memory is available for these tools.

Managing System Entropy

Entropy is used to calculate random data that is used by the system in cryptographic operations. Some environments with low entropy may have intermittent performance issues with SSL-based communication. This is more typical on virtual machines, but can occur in physical instances as well. Monitor the `kernel.random.entropy_avail` in `sysctl` value for best results.

If necessary, update `$JAVA_HOME/jre/lib/security/java.security` to use `file:/dev/./urandom` for the `securerandom.source` property.

Getting the Installation Packages

To begin the installation process, obtain the latest ZIP release bundle from Ping Identity and unpack it in a folder of your choice. The release bundle contains the Directory Proxy Server code, tools, and package documentation.

To Unpack the Build Distribution

1. Download the latest zip distribution of the Directory Proxy Server software.
2. Unzip the compressed zip archive file in a directory of your choice.

   ```
   $ unzip PingDirectoryProxy-<version>.zip
   ```

   You can now set up the Directory Proxy Server.

PingDirectoryProxy Server License Keys

License keys are required to install all PingDirectoryProxy Server products. Obtain licenses through Salesforce or from https://www.pingidentity.com/en/account/request-license-key.html.

- A license is always required for setting up a new single server instance and can be used site-wide for all servers in an environment. When cloning a server instance with a valid license, no new license is needed.
- A new license must be obtained when updating a server to a new major version, for example from 6.2 to 7.0. Licenses with no expiration date are valid until the server is upgraded to the next major version. A prompt for a new license is displayed during the update process.
- A license may expire on particular date. If a license does expire, obtain a new license and install it using `dsconfig` or the Administrative Console. The server will provide a notification as the expiration date approaches. License details are available using the server's `status` tool.

When installing the server, specify the license key file in one of the following ways:

- Copy the license key file to the server root directory before running setup. The interactive `setup` tool will discover the file and not require input. If the file is not in the server root, the `setup` tool will prompt for its location.
- If the license key is not in the server root directory, specify the `--licenseKeyFile` option for non-interactive setup, and the path to the file.

About the RPM Package

PingDirectoryProxy Server supports the PingDirectoryProxy Server release bundle in an RPM Package Manager (RPM) package for customers who require it. By default, the RPM unpacks the code at /opt/ping-identity/proxy/PingDirectoryProxy, after which you can run the `setup` command to install the server at that location.

If the RPM install fails for any reason, you can perform an RPM erase if the RPM database entry was created and manually remove the target RPM install directory (e.g., “/opt/ping-identity/proxy/PingDirectoryProxy” by default). You can install the package again once the system is ready.

To Install the RPM Package

1. Download the latest RPM distribution of the Directory Proxy Server software.
2. Unpack the build using the `rpm` command with the --install option. By default, the build is unpacked to `/opt/ping-identity/proxy/PingDirectoryProxy`. If you want to place the build at another location, use the `--prefix` option and specify the file path of your choice.

   ```
   $ rpm --install pingdirectoryproxy-<version>.rpm
   ```
3. From /opt/ping-identity/proxy/PingDirectoryProxy/PingDirectoryProxy, run the `setup` command to install the server on the machine.

Installing the Directory Proxy Server

When you deploy PingDirectoryProxy Server in a topology, you generally deploy them in pairs. These pairs are configured identically except for their host name, port name, and possibly their location.

To help administrators easily install identical proxies, the Directory Proxy Server allows you to clone a proxy configuration. First, you install a Directory Proxy Server using the `setup` tool. Then, you configure it using the `create-initial-proxy-config` tool described in *Using the create-initial-proxy-config Tool*. Finally, you run the `setup` tool on subsequent servers, indicating that you want to clone the configuration on a peer server.

The following sections describe the setup tool in more detail, and tell you how to install first and subsequent proxies in your topology.

About the setup Tool

One of the strengths of the PingDirectoryProxy Server is the ease with which you can install a server instance using the `setup` tool. The `setup` tool allows you to quickly install and configure a stand-alone Directory Proxy Server instance.

To install a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode prompts for information during the installation process. To run the installation in this mode, use the `setup --cli` command.
- **Non-Interactive Command-Line Mode**. Non-interactive command-line mode is designed for setup scripts to automate installations or for command-line usage. To run the installation in this mode, `setup` must be run with the --no-prompt option as well as the other arguments required to define the appropriate initial configuration.

All installation and configuration steps should be performed while logged on to the system as the user or role under which the Directory Proxy Server will run.

Installing the First Directory Proxy Server in Interactive Mode

The `setup` tool provides an interactive text-based interface to install a Directory Proxy Server instance.

To Install the First Directory Proxy Server in Interactive Mode

1. Change to the server root directory.

   ```
   cd Directory Proxy Server
   ```
2. Use the `setup` command.

   ```
   $ ./setup
   ```
3. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
4. Press **Enter** to accept the default of `no` in response to adding this new server to an existing topology.

   ```
   Would you like to add this server to an existing Directory Proxy Server
    topology? (yes / no) [no]:
   ```
5. Enter the fully qualified hostname for this server, or press **Enter** to accept the default.
6. Create the initial root user DN for this server, or press **Enter** to accept the default.
7. Enter and confirm a password for this account.
8. To enable the Directory Proxy Server services (Configuration, Documentation, and SCIM) and Administrative Console over HTTPS, press **Enter** to accept the default. After setup, individual services can be enabled or disabled by configuring the HTTPS Connection Handler.
9. Enter the port on which the Directory Proxy Server should accept connections from HTTPS clients, press **Enter** to accept the default.
10. Enter the port on which the Directory Proxy Server should accept connections from LDAP clients, press **Enter** to accept the default.
11. The next two options enable LDAPS and StartTLS. Press **Enter** to accept the default (yes), or type **no**. If either are enabled, certificate options are required. To use the Java Keystore or the PKCS#12 keystore, the keystore path and the key PIN are required. To use the PKCS#11 token, only the key PIN is required.
12. Choose a certificate server option:

    ```
    Certificate server options:
      1) Generate self-signed certificate (recommended for testing purposes
     only)
      2) Use an existing certificate located on a Java Keystore (JKS)
      3) Use an existing certificate located on a PKCS#12 keystore
      4) Use an existing certificate on a PKCS#11 token
    ```
13. Choose the desired encryption for backups and log files from the choices provided:

    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.
    - Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
    - Do not encrypt server data.
14. To configure your Directory Proxy Server to use entry balancing, type `yes`, or accept the default `no`. In an entry balancing environment, entries immediately beneath the balancing base DN are divided into disjoint subsets. Each subset of data is handled by a separate set of one or more directory server instances, which replicate this subset of data between themselves. Choosing `yes` will enable more memory be allocated to the server and tools.
15. Choose the option for the amount of memory to assign to this server.

16. Enter an option to setup the server with the current configurtion, provide new parameters, or cancel.
17. Once setup is complete, choose the next configuration option.

    ```
    This server is now ready for configuration What would you like to do?

       1) Start 'create-initial-proxy-config' to create a basic
          initial configuration (recommended for new users)
       2) Start 'dsconfig' to create a configuration from scratch
       3) Quit

    Enter choice [1]:
    ```

To Install Additional Directory Proxy Server Instances in Interactive Mode

The `setup` tool provides an interactive text-based interface to install a Directory Proxy Server instance that clones a previously installed Directory Proxy Server instance.

1. Change to the server root directory.

   ```
   cd Directory Proxy Server
   ```
2. Use the `setup` command.

   ```
   $ ./setup
   ```
3. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
4. Enter `yes` in response to add this new server to an existing topology.

   ```
   Would you like to add this server to an existing Directory Proxy Server
    topology? (yes / no) [no]: yes
   ```
5. Enter the host name of the Directory Proxy Server from which configuration settings are copied during setup.

   ```
   Enter the hostname of the peer Directory Proxy Server from which you would
    like
   to copy configuration settings. [proxy.example.com]:
   ```
6. Type the port number of the peer Directory Proxy Server from which configuration settings are copied during setup. You can press **Enter** to accept the default port, which is 389.

   ```
   Enter the port of the peer Directory Proxy Server [389]:
   ```
7. Enter the option corresponding to the type of connection you want to use to connect to the peer Directory Proxy Server.

   ```
   How would you like to connect to the peer Directory Proxy Server?
     1) None
     2) SSL
     3) StartTLS

   Enter choice [1]:
   ```
8. Type the root user DN of the peer Directory Proxy Server, or press **Enter** to accept the default (cn=Directory Manager), and then type and confirm the root user password.

   ```
   Enter the manager account DN for the peer Directory Proxy Server
    [cn=Directory Manager]:
   Enter the password for cn=Directory Manager:
   ```
9. Enter the host name of the new local Directory Proxy Server.

   ```
   Enter the fully qualified host name or IP address of the local host
    [proxy.example.com]:
   ```
10. Choose the location of your new Directory Proxy Server instance or enter a new one.
11. Enter an option to setup the server with the current configurtion, provide new parameters, or cancel.

12. Once setup is complete, choose the next configuration option.

Installing the First Directory Proxy Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. If there is a missing or incorrect argument, the `setup` tool fails and aborts the process.

The `setup` tool automatically chooses the maximum heap size. You can manually tune the maximum amount of memory devoted to the server's process heap using the --maxHeapSize option. The --maxHeapSize argument is only valid if the --entryBalancing or --aggressiveJVMTuning options are also present.

For example, use the --aggressiveJVMTuning option to set the maximum amount of memory used by the Directory Proxy Server and tools as follows:

```
--aggressiveJVMTuning --maxHeapSize 256m
```

If you are using entry balancing, tune the amount of memory devoted to the Directory Proxy Server using the --entryBalancing option as follows:

```
--entryBalancing --maxHeapSize 1g
```

The amount of memory allowed when using the --entryBalancing option is calculated and depends on the amount of system memory available. If you are using entry balancing and also want the tools to get more memory, include both the --entryBalancing and the --aggressiveJVMTuning options.

```
--entryBalancing --aggressiveJVMTuning --maxHeapSize 1g
```

If you have already configured a truststore, you can also use the setup tool to enable security. The following example enables security, both SSL and StartTLS. It also specifies a JKS Keystore and Truststore that define the server certificate and trusted CA. The passwords for the keystore files are defined in the corresponding .pin files, where the password is written on the first line of the file. The values in the .pin files will be copied to the `server-root/config` directory in the `keystore.pin` file.

Note that the password to the private key within the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined within the Administrative Console or `dsconfig` by editing the Trust Manager Provider standard configuration object.

```
$ env JAVA_HOME=/ds/java ./setup --cli \
   --no-prompt --rootUserDN "cn=Directory Manager" \
   --rootUserPassword "password" --ldapPort 389 \
   --enableStartTLS --ldapsPort 636 \
   --useJavaKeystore /path/to/devkeystore.jks \
   --keyStorePasswordFile /path/to/devkeystore.pin \
   --certNickName server-cert \
   --useJavaTrustStore /path/to/devtruststore.jks \
   --trustStorePasswordFile /path/to/devtruststore.pin \
   --acceptLicense
```

To Install the First Directory Proxy Server in Non-Interactive Mode

- Use `setup` with the --no-prompt option. The command uses the default root user DN (`cn=Director Manager`) with the specified --rootUserPassword option. You must include the --acceptLicense option or the `setup` tool will generate an error message.

  ```
  $ env JAVA_HOME=/ds/java ./setup --no-prompt \
  --rootUserDN "cn=Directory Manager" \
  --rootUserPassword "password" --ldapPort 389 \
  --acceptLicense
  ```

To Install Additional Directory Proxy Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. If there is a missing or incorrect argument, the `setup` tool fails and aborts the process.

To Install Additional Directory Proxy Server in Non-Interactive Mode

- Use `setup` with the --no-prompt option.

```
$ env JAVA_HOME=/ds/java ./setup --cli --no-prompt \
--rootUserDN "cn=Directory Manager" \
--rootUserPassword "password" --ldapPort 1389 \
--localHostName proxy2.example.com \
--peerHostName proxy1.example.com --peerPort 389 \
--peerUseNoSecurity --acceptLicense --location austin1
```

Installing the Directory Proxy Server with a Truststore in Non-Interactive Mode

If you have already configured a trust store, you can also use the `setup` tool to enable security. The following example enables SSL security. It also specifies a JKS Keystore and truststore that define the server certificate and trusted CA. The passwords for the keystore files are defined in the corresponding .pin files, where the password is written on the first line of the file. The values in the .pin files will be copied to the `server-root/config` directory in the `keystore.pin` and `truststore.pin` files.

Note: The password to the private key within the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined within the Administrative Console or `dsconfig` by editing the Key Manager Provider standard configuration object.

To Install the Directory Proxy Server with a Truststore in Non-Interactive Mode

- Run the `setup` tool to install a Directory Proxy Server with a truststore.

```
$ env JAVA_HOME=/ds/java ./setup --cli \
  --no-prompt --rootUserDN "cn=Directory Manager" \
  --rootUserPassword "password" \
  --ldapPort 389 --ldapsPort 636 \
  --useJavaKeystore /path/to/devkeystore.jks \
  --keyStorePasswordFile /path/to/devkeystore.pin \
  --certNickName server-cert \
  --useJavaTrustStore /path/to/devtruststore.jks \
  --acceptLicense

In order to update the trust store, the password must be provided

See 'prepare-external-server --help' for general overview

Testing connection to ds-east-01.example.com:1636 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access .....
Created 'cn=Proxy User,cn=Root DNs,cn=config'

Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

About the Layout of the Directory Proxy Server Folders

Once you have unzipped the Directory Proxy Server distribution file, the following folders and command-line utilities are available.

Table 1: Layout of the Directory Proxy Server Folders

Directories/Files/Tools	Description
License.txt	Licensing agreement for the Directory Proxy Server.
README	README file that describes the steps to set up and start the Directory Proxy Server.
bak	Stores the physical backup files used with the `backup` command-line tool.
bat	Stores Windows-based command-line tools for the Directory Proxy Server.
bin	Stores UNIX/Linux-based command-line tools for the Directory Proxy Server.
classes	Stores any external classes for server extensions.
collector	Used by the server to make monitored statistics available to the Data Metrics Server.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
docs	Provides the product documentation.
import-tmp	Stores temporary imported items.
ldif	Stores any LDIF files that you may have created or imported.
legal-notices	Stores any legal notices for dependent software used with the Directory Proxy Server.
lib	Stores any scripts, jar, and library files needed for the server and its extensions.
locks	Stores any lock files in the backends.
logs	Stores log files for the Directory Proxy Server.
metrics	Stores the metrics that can be gathered for this server and surfaced in the Data Metrics Server.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
revert-update	The `revert-update` tool for UNIX/Linux systems.
revert-update.bat	The `revert-update` tool for Windows systems.
setup	The `setup` tool for UNIX/Linux systems.
setup.bat	The `setup` tool for Windows systems.
scim-data-tmp	Used to create temporary files containing SCIM request data.
uninstall	The `uninstall` tool for UNIX/Linux systems.
uninstall.bat	The `uninstall` tool for Windows systems.
update	The `update` tool for UNIX/Linux systems.
update.bat	The `update` tool for Windows systems.
Velocity	Stores any customized Velocity templates and other artifacts (CSS, Javascript, images), or Velocity applications hosted by the server.

Running the Server

To start the Directory Proxy Server, run the bin/start-server command on UNIX or Linux systems (an analogous command is in the `bat` folder on Microsoft Windows systems). The bin/start-server command starts the Directory Proxy Server as a background process when no options are specified. To run the Directory Proxy Server as a foreground process, use the bin/start-server command with the --nodetach option.

To Start the Directory Proxy Server

Use bin/start-server to start the server.

```
$ bin/start-server
```

To Run the Server as a Foreground Process

1. Enter bin/start-server with the --nodetach option to launch the Directory Proxy Server as a foreground process.

```
$ bin/start-server --nodetach
```

2. You can stop the Directory Proxy Server by pressing `CNTRL+C` in the terminal window where the server is running or by running the bin/stop-server command from another window.

To Start the Server at Boot Time

By default, the PingDirectoryProxy Server does not start automatically when the system is booted. Instead, you must manually start it with the bin/start-server command. To configure the Directory Proxy Server to start automatically when the system boots, use the `create-systemd-script` utility to create a script, or create the script manually.

1. Create the service unit configuration file in a temporary location where "ds" is the user the PingDirectoryProxy will run as.

```
$ bin/create-systemd-script \
     --outputFile /tmp/ping-directory.service \
     --userName ds
```

2. As a root user, copy the `ping-directory.service` configuration file into the `/etc/systemd/system` directory.
3. Reload `systemd` to read the new configuration file.

```
$ systemctl daemon-reload
```

4. To start the PingDirectoryProxy, use the `start` command.

```
$ systemctl start ping-directory.service
```

5. To configure the PingDirectoryProxy to start automatically when the system boots, use the `enable` command.

```
$ systemctl enable ping-directory.service
```

6. Log out as root.

   If on an RC system, this task is done by creating the startup script with `bin/create-rc-script` and moving it to the `/etc/init.d` directory. Create symlinks to it from the `/etc/rc3.d` directory (staring with an “S” to ensure that the server is started) and `/etc/rc0.d` directory (starting with a “K” to ensure that the server is stopped).

Logging into the Administrative Console

After the server is installed, access the Administrative Console, `https://hostname:HTTPport/console/login`, to verify the configuration and manage the server. To log into the Administrative Console, use the initial root user DN specified during setup (by default `cn=Directory Manager`).

The `dsconfig` command or the Administrative Console can be used to create additional root DN users in `cn=Root DNs,cn=config`. These new users require the fully qualified DN as the login name, such as `cn=new-admin,cn=Root DNs,cn=config`. To use a simple user name (with out the `cn=` prefix) for logging into the Administrative Console, the root DN user must have the `alternate-bind-dn` attribute configured with an alternate name, such as "admin."

By default the link to the Administrative Console is `https://hostname:HTTPport/console/login`.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package (`/server-root/resource/admin-console.zip`) can be installed according to that container's documentation.

Stopping the Directory Proxy Server

The Directory Proxy Server provides a simple shutdown script, bin/stop-server, to stop the server. You can run it manually from the command line or within a script.

If the Directory Proxy Server has been configured to use a large amount of memory, then it can take several seconds for the operating system to fully release the memory and make it available again. If you try to start the server too quickly after shutting it down, then the server can fail because the system does not yet have enough free memory. On UNIX systems, run the `vmstat` command and watch the values in the "free" column increase until all memory held by the Directory Proxy Server is released back to the system.

You can also set a configuration option that specifies the maximum shutdown time a process may take.

To Stop the Server

- Use the bin/stop-server tool to shut down the server.

```
$ bin/stop-server
```

To Schedule a Server Shutdown

- Use the `bin/stop-server` tool with the --stopTime YYYYMMDDhhmmss option to schedule a server shutdown.

  The Directory Proxy Server schedules the shutdown and sends a notification to the `server.out` log file. The following example sets up a shutdown task that is scheduled to be processed on June 6, 2012 at 8:45 A.M. CDT. The server uses the UTC time format if the provided timestamp includes a trailing “Z”, for example, 20120606134500Z. The command also uses the --stopReason option that writes the reason for the shut down to the logs.

```
$ bin/stop-server --stopTime 20120606134500Z --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --stopReason "Scheduled offline maintenance"
```

To Restart the Server

Re-start the Directory Proxy Server using the bin/stop-server command with the `--restart` or `-R` option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again.

- Go to the server root directory, and run the bin/stop-server command with the -R or --restart options.

```
$ bin/stop-server --restart
```

Run the Server as a Microsoft Windows Service

The server can run as a Windows service on Windows Server 2012 R2 and Windows Server 2016. This enables log out of a machine without the server being stopped.

To Register the Server as a Windows Service

Perform the following steps to register the server as a service:

1. Stop the server with `bin/stop-server`. A server cannot be registered while it is running.
2. Register the server as a service. From a Windows command prompt, run `bat/register-windows-service.bat`.
3. After a server is registered, start the server from the Windows Services Control Panel or with the `bat/start-server.bat` command.

Command-line arguments for the `start-server.bat` and `stop-server.bat` scripts are not supported while the server is registered to run as a Windows service. Using a task to stop the server is also not supported.

To Run Multiple Service Instances

Only one instance of a particular service can run at one time. Services are distinguished by the `wrapper.name` property in the `<server-root>/config/wrapper-product.conf` file. To run additional service instances, change the `wrapper.name` property on each additional instance. Descriptions of the services can also be added or changed in the `wrapper-product.conf` file.

To Deregister and Uninstall Services

While a server is registered as a service, it cannot run as a non-service process or be uninstalled. Use the `bat/deregister-windows-service.bat` file to remove the service from the Windows registry. The server can then be uninstalled with the `uninstall.bat` script.

Log Files for Services

The log files are stored in `<server-root>/logs`, and filenames start with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or due to file size. Only the last three log files are retained. These configurations can be changed in the `<server-root>/config/wrapper.conf` file.

Uninstalling the Server

The Directory Proxy Server provides an `uninstall` command-line utility for quick and easy removal of the code base.

To uninstall a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode is a text-based interface that prompts the user for input. You can start the command using the `bin/uninstall` command with the --cli option. The utility prompts you for input if more data is required.
- **Non-Interactive Command-Line Mode**. Non-interactive mode suppresses progress information from being written to standard output during processing, except for fatal errors. This mode is convenient for scripting and is invoked using the `bin/uninstall` command with the --no-prompt option.

Note: For stand-alone installations with a single Directory Proxy Server instance, you can also manually remove the Directory Proxy Server by stopping the server and recursively deleting the directory and subdirectories. For example:

```
$ rm -rf /ds/PingDirectoryProxy
```

To Uninstall the Server in Interactive Mode

Interactive mode uses a text-based, command-line interface to help you remove your instance. If `uninstall` cannot remove all of the Directory Proxy Server files, the `uninstall` tool generates a message with a list of the files and directories that must be manually deleted. The `uninstall` command must be run as either the root user or the same user (or role) that installed the Directory Proxy Server.

1. From the server root directory, run the `uninstall` command.

```
$ ./uninstall --cli
```

2. Select the components to be removed. If you want to remove all components, press **Enter** to accept the default (remove all). Enter the option to specify the specific components that you want to remove.

```
Do you want to remove all components or select the components to remove?
```

```
1) Remove all components
2) Select the components to be removed

q) quit
Enter choice [1]:
```

3. For each type of server component, press **Enter** to remove them or type `no` to keep it.

```
Remove Server Libraries and Administrative Tools? (yes / no) [yes]:
Remove Database Contents? (yes / no) [yes]:
Remove Log Files? (yes / no) [yes]:
Remove Configuration and Schema Files? (yes / no) [yes]:
Remove Backup Files Contained in bak Directory? (yes / no) [yes]:
Remove LDIF Export Files Contained in ldif Directory? (yes / no) [yes]:
```

4. If the Directory Proxy Server is part of a replication topology, type `yes` to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a stand-alone server, continue to step 7.
5. Type the Global Administrator ID and password to remove the references to this server in other replicated servers. Then, type or verify the host name or IP address for the server that you are uninstalling.
6. Next, select how you want to trust the server certificate if you have set up SSL or StartTLS. For this example, press **Enter** to accept the default.

```
How do you want to trust the server certificate for the Directory Proxy
 Server
on server.example.com:389?

1) Automatically trust
2) Use a trust store
3) Manually validate

Enter choice [3]:
```

7. If your Directory Proxy Server is running, the server is shutdown before continuing the uninstall process. The uninstall processes the removal requests and completes. View the logs for any remaining files. Manually remove any remaining files or directories, if listed.

To Uninstall the Server in Non-Interactive Mode

The `uninstall` utility provides a non-interactive method to enter the command with the --no-prompt option. Another useful argument is the --forceOnError option that continues the uninstall process when an error is encountered. If an option is incorrectly entered or if a required option is omitted and the --forceOnError option is not used, the command will fail and abort.

1. From the server root directory, run `uninstall` tool with the --remove-all option to remove all of the Directory Proxy Server's libraries. The --quiet option suppresses output information and is optional. The following command assumes that the Directory Proxy Server is stand-alone and not part of a replication topology.

```
$ ./uninstall --cli --remove-all --no-prompt --quiet --forceOnError
```

2. If any files or directories remain, manually remove them.

To Uninstall Selected Components in Non-Interactive Mode

From the server root directory, run `uninstall` with the --backup-files option to remove the Directory Proxy Server's backup files. Use the --help or -H option to view the other options available to remove specific components.

```
$ ./uninstall --cli --backup-files --no-prompt --quiet --forceOnError
```

To Uninstall the RPM Build Package

1. From the server root directory, remove the RPM package use the --erase option with the <rpm-id>. The <rpm-id> is pingdirectoryproxy and removes the files at /opt/ping-identity/proxy/PingDirectoryProxy/PingDirectoryProxy.

```
$ rpm --erase pingdirectoryproxy
```

2. The `rpm` command specifies if any files or directories require manual deletion. Manually remove any remaining directories or files using `rm -rf <directory>`.

Updating the Directory Proxy Server

Ping Identity issues new software builds periodically and distributes the software package in zip format. Administrators can use the Directory Proxy Server's `update` utility to update the current server code with the latest features and bug fixes. To update the Directory Proxy Server to a newer version, download the build package, and then unzip the new server package on the same host as the server that you wish to update. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors.

During an update process, the updater checks a manifest file that contains a MD5 checksum of each file in its original state when installed from zip. Next, it compares the checksum of the new server files to that of the old server. Any files that have different checksums will be updated. For files that predates the manifest file generation, the file is backed up and replaced. The updater also logs all file changes in the history directory to tell what files have been changed.

For schema updates, the `update` tool preserves any custom schema definitions (`99-user.ldif`). For any default schema element changes, if any, the updater will warn the user about this condition and then create a patch schema file and copy it into the server's schema directory. For configuration files, the update tool preserves the configuration file, `config.ldif`, unless new configuration options must be added to the Directory Proxy Server.

Once the updater finishes its processing, it checks if the newly updated server starts without any fatal errors. If an error occurs during the update process, the `update` tool reverts the server root instance to the server state prior to the update.

About Updating Servers in a Topology

An update to the current release includes significant changes, and the introduction of a topology registry, which will store information previously stored in the admin backend (server instances, instance and secret keys, server groups, and administrator user accounts). For the admin backend to be migrated, the `update` tool must be provided LDAP authentication options to the peer servers of the server being updated.

The LDAP connection security option requested (either plain, TLS, StartTLS, or SASL) must be configured on every server in the topology. The LDAP credentials must be present on every server in the topology, and must have permissions to read from the admin backend and the config backend of every server in the topology. For example, a root DN user with the `inherit-default-privileges` set to true (such as the `cn=Directory Manager` user) that exists on every server can be used.

After enabling or fixing the configuration of the LDAP connection handler(s) to support the desired connection security mechanism on each server, run the following `dsframework` command on the server being updated so that its admin backend has the most up-to-date information:

```
$ bin/dsframework set-server-properties \
  --serverID serverID \
  --set ldapport:port \
  --set ldapsport:port \
  --set startTLSEnabled:true
```

The `update` tool will verify that the following conditions are satisfied on every server in the topology before allowing the update:

- When the first server is being updated, all other servers in the topology must be online. When updating additional servers, all topology information will be obtained from one of the servers that has already been updated. The `update` tool will connect to the peer servers of the server being updated to obtain the necessary information to populate the topology registry. The provided LDAP credentials must have read permissions to the config and admin backends of the peer servers.

- The instance name is set on every server, and is unique across all servers in the topology. The instance name is a server's identifier in the topology. After all servers in the topology have been updated, each server will be uniquely identified by its instance name. Once set, the name cannot be changed. If needed, the following command can be used to set the instance name of a server prior to the update:

```
$ bin/dsconfig set-global-configuration-prop \
  --set instance-name:uniqueName
```

- The cluster-wide configuration is synchronized on all servers in the topology. Older versions have some topology configuration under `cn=cluster,cn=config` (JSON attribute and field constraints). These items did not support mirrored cluster-wide configuration data. An update should avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. To synchronize the cluster-wide configuration data across all servers in the topology, run the `config-diff` tool on each pair of servers to determine the differences, and use `dsconfig` to update each instance using the `config-diff` output. For example:

```
$ bin/config-diff --sourceHost hostName \
  --sourcePort port \
  --sourceBindDN bindDN \
  --sourceBindPassword password \
  --targetHost hostName \
  --targetPort port \
  --targetBindDN bindDN \
  --targetBindPassword password
```

If any of these conditions are not satisfied, the `update` tool will list all of the errors encountered for each server, and provide instructions on how to fix them.

To Update the Directory Proxy Server

Assume that an existing version of the Directory Proxy Server is stored at PingDirectoryProxy-old, which you want to update.

1. Make sure you have complete, readable backup of the existing system before upgrading the Directory Proxy Server build. Also, make sure you have a clear backout plan and schedule.
2. Download the latest version of the PingDirectoryProxy Server software and unzip the file. For this example, let's assume the new server is located in the PingDirectoryProxy-new directory.
3. Check the version number of the newly downloaded Directory Proxy Server instance using the --version option on any command-line utility. For example, you should see the latest revision number.

```
$ PingDirectoryProxy-new/setup --version PingDirectoryProxy Server 7.0.1.0
Build 20110432006092 Revision 9235
```

4. Use the `update` tool of the newly unzipped build to update the Directory Proxy Server code. Make sure to specify the Directory Proxy Server instance that you are upgrading with the --serverRoot option. The Directory Proxy Server must be stopped for this update to be applied.

```
$ PingDirectoryProxy-new/update --serverRoot PingDirectoryProxy-old
```

Note: The PingDirectoryProxy Server provides a web console called the Administrative Console, to configure and monitor the server. If you update the Directory Proxy Server version, you should also update the Administrative Console.

5. View the log file to see which files were changed. The log file is located in the `<server-root>/history` directory. For example, the file will be labelled with the Directory Proxy Server version number and revision.

```
$ view <server-root>/history/1272307020420-7.0.1.0.9235/update.log
```

To Upgrade the RPM Package

If the Linux RPM package was used to install the Directory Server, the following should be performed to upgrade the server.

- Assume that the new RPM package, pingdirectoryproxy-<new-version>.rpm, is placed in the server root directory. From the server root directory, run the `rpm` command with the --upgrade option.

  ```
  $ rpm --upgrade pingdirectoryproxy-<new-version>.rpm
  ```

  The RPM package does not support a revert option once the build is upgraded.

  The upgrade history is written to /opt/ping-identity/proxy/PingDirectoryProxy/PingDirectoryProxy/history/<timestamp>/update.log.

Reverting an Update

Once the Directory Proxy Server has been updated, you can revert to the most recent version (one level back) using the `revert-update` tool. The `revert-update` tool accesses a log of file actions taken by the updater to put the filesystem back to its prior state. If you have run multiple updates, you can run the `revert-update` tool multiple times to revert to each prior update sequentially. You can only revert back one level. For example, if you have run the update twice since first installing the Directory Proxy Server, you can run the `revert-update` command to revert to its previous state, then run the `revert-update` command again to return to its original state.

Note: Reverting an update is not supported for upgrades to version 7.0, due to the topology backend changes.

To Revert to the Most Recent Server Version

Use `revert-update` in the server root directory revert back to the most recent version of the server.

```
$ PingDirectoryProxy-old/revert-update
```

Configure SCIM After Upgrade

Modifications in SCIM PATCH are mapped directly to LDAP modifications to use the matching rules configured in the Directory Proxy Server, when matching deleted values. Since the SCIM PATCH is now applied by the Directory Server, the Permissive Modify Request Control (1.2.840.113556.1.4.1413) is now required by the SCIM component. This ensures that adding an existing value or deleting a non-existent value in the PATCH request will not generate an error. This affects upgrades from server versions prior to 5.0.0.

To continue using the SCIM component after an upgrade, access controls and configuration must be updated to allow access to the Permissive Modify Request Control. Run the `dsconfig` commands to update these components:

```
$ dsconfig set-access-control-handler-prop \
  --remove 'global-aci:(targetcontrol="1.3.6.1.1.13.2 ||
 1.2.840.113556.1.4.473 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9
 || 1.3.6.1.1.12")(version 3.0;acl "Authenticated access to controls used by
 the SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473
 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9 || 1.3.6.1.1.12 ||
 1.2.840.113556.1.4.1413")(version 3.0;acl "Authenticated access to controls
 used by the SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

```
dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add supported-control-oid:1.2.840.113556.1.4.1413
```

In the last command, `dc_example_dc_com-req-processor` is the default processor name. Replace it with the correct name for your system.

Chapter

3

Configuring the Directory Proxy Server

Topics:

Once you have initially configured the PingDirectoryProxy Server, you can manage your deployment using the configuration framework and management tools. This chapter briefly describes these tools and provides procedures to help you maintain and update your deployment.

It includes the following sections:

- *Configuring the Global Referential Integrity Plug-in*
- *Configuring an Active Directory Server Back-end*

About the Configuration Tools

The PingDirectoryProxy Server configuration can be accessed and modified in the following ways:

- **Using the Administrative Console**. The PingDirectoryProxy Server provides an Administrative Console for graphical server management and monitoring. The console provides equivalent functionality as the `dsconfig` command for viewing or editing configurations. All configuration changes using this tool are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands should you need to back out of a configuration.
- **Using the dsconfig Command-Line Tool**. The `dsconfig` tool is a text-based menu-driven interface to the underlying configuration. The tool runs the configuration using three operational modes: interactive command-line mode, non-interactive command-line mode, and batch mode. All configuration changes made using this tool are recorded in `logs/config-audit.log`.

Using the create-initial-proxy-config Tool

The `create-initial-proxy-config` tool can be used to initially configure the Directory Proxy Server. We strongly recommend that you use the `create-initial-proxy-config` tool for your initial Directory Proxy Server configuration. This tool prompts you for basic information about your topology, including external servers, their locations, and credentials for communicating with them. Once the configuration is complete, the tool writes the configuration to a `dsconfig` batch file and allows you to apply the configuration to the local Directory Proxy Server. The tool assumes the following about your topology:

- All servers are accessible through a single user account. This user account must be a root user that is not generally accessible to clients to avoid inadvertent changes, deletions, or backend server availability issues due to reimporting data.
- All servers support the same type of communication security.
- All external servers are any combination of Ping Identity Directory Server, Sun Directory Server, or Red Hat (including Fedora and 389) instances.

If your topology does have these characteristics, you can use the tool to define a basic configuration that is saved to a `dsconfig` batch file. You can then run the `dsconfig` tool to fine-tune the configuration. You can also use this tool to configure an entry balancing configuration, which allows you to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance.

The `create-initial-proxy-config` tool produces a log file called `create-initial-proxy-config.log` that is stored in the local Directory Proxy Server's `logs` directory.

You can only run the `create-initial-proxy-config` tool once for the initial configuration of each Directory Proxy Server instance. To tune your configuration, use the `dsconfig` tool. When installing a second Directory Proxy Server, it will not be necessary to run the `create-initial-proxy-config` tool again, as the Directory Proxy Server setup has the ability to clone the settings from an existing Directory Proxy Server.

This section describes how to use this tool to configure a standard Directory Proxy Server deployment as well as an entry balancing configuration.

Configuring a Standard Directory Proxy Server Deployment

This section describes how to install a standard Directory Proxy Server deployment using the `create-initial-proxy-config` tool. Remember that you deploy the Directory Proxy Server in pairs. Each pair should be configured identically except for their host name, port, and possibly their location.

To Configure a Standard Directory Proxy Server Deployment

1. After initial installation, select the number to start the `create-initial-proxy-config` tool automatically. Otherwise, run it manually at the command line from the server root directory, <server-root>/PingDirectoryProxy.

```
$ ./bin/create-initial-proxy-config
```

2. The initial proxy configuration presents the assumptions about the underlying Directory Server backend servers. If the servers do not meet the requirements, then you can enter "no" to quit the process.

```
Some assumptions are made about the topology in order to keep this tool
 simple:

 1) all servers will be accessible via a single user account
 2) all servers support the same communication security type
 3) all servers are PingDirectoryProxy Server, Directory Server,
    Java System 5.x, 6.x, or 7.x, or Red Hat (including Fedora and 389)
    directory servers

If your topology does not have these characteristics you can use this tool
 to define a basic configuration and then use the 'dsconfig' tool or the
 Administrative Console to
fine tune the configuration.

Continue? (yes / no) [yes]:
```

3. Enter the DN for the Directory Proxy Server user account, then enter and confirm the password for this account. Note that you should not use `cn=Directory Manager` account for communication between the Directory Proxy Server and the Directory Server. For security reasons, the account used to communicate between the Directory Proxy Server and the Directory Server should not be directly accessible by clients accessing the Directory Proxy Server. For more information about this account, see *Configuring LDAP External Servers*.

```
Enter the DN of the proxy user account [cn=Proxy User,cn=Root
 DNs,cn=config]:
Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

4. Specify whether you will be using secure communication with the Directory Server instances.

```
>>>> External Server Communication Security

Specify the type of security that the Directory Proxy Server will use when
 communicating with directory server instances:

   1)  None
   2)  SSL
   3)  StartTLS

   b)  back
   q)  quit

Enter choice [1]:
```

5. Specify the base DNs of the Directory Server instances that will be accessed through the Directory Proxy Server. The Directory Proxy Server will create subtree views using each base DN to define portions of the external servers' DIT available for client access. You can specify more than one base DN. Press **Enter** when you have finished specifying the DN(s).

```
Enter a base DN of the directory server instances that will be accessed
 through the Identity Proxy:

    b)  back
    q)  quit

Enter a DN or choose a menu item [dc=example,dc=com]:
```

6. Next, specify if the entries under your defined subtree view will be split across multiple servers in an entry balanced deployment. For this example, press Enter to accept the default ("no").

7. Define a location for your server, such as the name of your data center or the city where the server is located. This example illustrates defining a location named `east`.

   ```
   Enter a location name or choose a menu item: east
   ```

8. If you defined more than one location, specify the location that contains the Directory Proxy Server itself.

   ```
   Choose the location for this Directory Proxy Server

       1)  east
       2)  west

       b)  back
       q)  quit

   Enter choice [1]: 1
   ```

9. Define the hostname:port used by the LDAP external servers. If you have specified more than one location, you will go through this process for each location.

   ```
   Enter a host:port or choose a menu item [localhost:389]: ldap-
   east-01.example.com:389
   ```

10. After each step, the server will attempt to prepare each external server by testing the communication between the Directory Proxy Server and the Directory Server. Select the option "Yes, and all subsequent servers" to indicate that you want the tool to create a proxy user account on all of your LDAP external servers within that location.

    ```
    Would you like to prepare ldap-east-01.example.com:389 for access by the
     Directory Proxy Server?

        1) Yes
        2) No
        3) Yes, and all subsequent servers
        4) No, and all subsequent servers

    Enter choice [1]: 3
    ```

11. If the proxy user account did not previously exist on your LDAP external server, create the account by connecting as `cn=Directory Manager`.

    ```
    Would you like to create or modify root user 'cn=Proxy User' so that it is
     available for this Directory Proxy Server? (yes / no) [yes]:
    Enter the DN of an account on ldap-east-01.example.com:389 with which to
     create or manage the 'cn=Proxy
    User' account [cn=Directory Manager]:

    Enter the password for 'cn=Directory Manager':

    Created 'cn=Proxy User,cn=Root DNs,cn=config'
    Testing 'cn=Proxy User' privileges ..... Done
    Verifying backend 'dc=example,dc=com' ..... Done
    ```

12. Repeat steps 9-12 for the servers in the other location. Then, press **Enter** to finish configuring the location.

13. Review the configuration summary. Once you have confirmed that the changes are correct, press **Enter** to write the configuration.

    ```
    >>>> Configuration Summary

      External Server Security:  SSL
      Proxy User DN:             cn=Proxy User,cn=Root DNs,cn=config

      Location east
        Failover Order: west
        Servers: localhost:1636
    ```

```
  Location west
    Failover Order: east
    Servers: localhost:2636

  Base DN: dc=example,dc=com
    Servers: localhost:1636, localhost:2636


    b)  back
    q)  quit
    w)  write configuration file

Enter choice [w]:
```

14. Next, apply the configuration changes locally to the Directory Proxy Server. If you have any Server SDK extensions, make sure to run the `manage-extension` tool, then press **Enter** to apply the changes to the Directory Proxy Server. Alternatively, you can quit and instead run the `dsconfig` batch file at a later time. Once the changes have been applied, you cannot use the `create-initial-proxy-config` tool to configure this Directory Proxy Server again. Instead, use the `dsconfig` tool.

```
This tool can apply the configuration changes to the local Identity Proxy.
 This requires any configured Server SDK extensions to be in place. Do you
 want to do
this? (yes / no) [yes]:
```

If you open the generated `proxy-cfg.txt` file or the `logs/config-audit.log` file, you will see that a configuration element hierarchy has been created: locations, health checks, external servers, load-balancing algorithms, request processors, and subtree views.

About dsconfig Configuration Tool

The `dsconfig` tool is the text-based management tool used to configure the underlying Directory Server configuration. The tool has three operational modes: interactive mode, non-interactive mode, and batch mode.

The `dsconfig` tool also offers an offline mode using the --offline option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Using dsconfig in Interactive Command-Line Mode

In interactive mode, the `dsconfig` tool offers a filtering mechanism that only displays the most common configuration elements. The user can specify that more expert level objects and configuration properties be shown using the menu system.

Running `dsconfig` in interactive command-line mode provides a user-friendly, menu-driven interface for accessing and configuring the PingDirectoryProxy Server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` script without any arguments. You will be prompted for connection and authentication information to the Directory Proxy Server, and then a menu will be displayed of the available operation types.

In some cases, a default value will be provided in square brackets. For example, [389] indicates that the default value for that field is port 389. You can press **Enter** to accept the default. To skip the connection and authentication prompts, provide this information using the command-line options of `dsconfig`.

Using dsconfig Interactive Mode: Viewing Object Menus

Because some configuration objects are more likely to be modified than others, the PingDirectoryProxy Server provides four different object menus that hide or expose configuration objects to the user. The purpose of object levels is to simply present only those properties that an administrator will likely use. The Object type is a convenience feature designed to unclutter menu readability.

The following object menus are available:

- **Basic**. Only includes the components that are expected to be configured most frequently.
- **Standard**. Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.
- **Advanced**. Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be potentially harmful if configured incorrectly.
- **Expert**. Includes all components in the Basic, Standard, and Advanced menus plus other components that should almost never require configuration or that could seriously impact the functionality of the server if not properly configured.

To Change the dsconfig Object Menu

1. Repeat steps 1–6 in the section using `dsconfig` in To Install the Directory Proxy Server in Interactive Mode.
2. On the **PingDirectoryProxy Server configuration** main menu, type **o** (letter "o") to change the object level. By default, Basic objects are displayed.
3. Enter a number corresponding to a object level of your choice: `1` for Basic, `2` for Standard, `3` for Advanced, `4` for Expert.
4. View the menu at the new object level. Additional configuration options for the Directory Proxy Server components are displayed.

Using dsconfig in Non-Interactive Mode

The `dsconfig` non-interactive command-line mode provides a simple way to make arbitrary changes to the Directory Proxy Server by invoking it from the command line. To use administrative scripts to automate configuration changes, run the `dsconfig` command in non-interactive mode, which is convenient scripting applications. Note, however, that if you plan to make changes to multiple configuration objects at the same time, then the batch mode might be more appropriate.

You can use the `dsconfig` tool to update a single configuration object using command-line arguments to provide all of the necessary information. The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The --no-prompt argument indicates that you want to use non-interactive mode. The {sub-command} is used to indicate which general action to perform. The {globalArgs} argument provides a set of arguments that specify how to connect and authenticate to the Directory Proxy Server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on your setup. For example, using standard LDAP connections, you can invoke the `dsconfig` tool as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

If your system uses SASL GSSAPI (Kerberos), you can invoke `dsconfig` as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The {subcommandArgs} argument contains a set of arguments specific to the particular subcommand that you wish to invoke. To always display the advanced properties, use the --advanced command-line option.

Note: Global arguments can appear anywhere on the command line (including before the subcommand, and after or intermingled with subcommand-specific arguments). The subcommand-specific arguments can appear anywhere after the subcommand.

To Get the Equivalent dsconfig Non-Interactive Mode Command

1. Using `dsconfig` in interactive mode, make changes to a configuration but do not apply the changes (that is, do not enter "f").
2. Enter `d` to view the equivalent non-interactive command.
3. View the equivalent command (seen below), and then press **Enter** to continue. For example, based on an example in the previous section, changes made to the `db-cache-percent` returns the following:

```
Command line to apply pending changes to this Local DB Backend:
dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:40
```

   The command does not contain the LDAP connection parameters required for the tool to connect to the host since it is presumed that the command would be used to connect to a different remote host.

Using dsconfig Batch Mode

The PingDirectoryProxy Server provides a `dsconfig` batching mechanism that reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The --no-prompt option is required with `dsconfig` in batch mode.

If a `dsconfig` command has a missing or incorrect argument, the command will fail and abort the batch process without applying any changes to the Directory Proxy Server. The `dsconfig` command supports a `--batch-continue-on-error` option which instructs `dsconfig` to apply all changes and skip any errors.

You can view the `logs/config-audit.log` file to review the configuration changes made to the Directory Proxy Server and use them in the batch file. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The Directory Proxy Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to PingDirectoryProxy Server machines.

To Configure the Directory Proxy Server in dsconfig Batch Mode

1. Create a text file that lists each `dsconfig` command with the complete set of properties that you want to apply to the Directory Proxy Server. The items in this file should be in the same format as those accepted by the `dsconfig` command. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

```
# This dsconfig operation creates the exAccountNumber global attribute
 index.
dsconfig create-global-attribute-index
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--index-name exAccountNumber --set prime-index:true

# Here we create the entry-count placement algorithm with the
# default behavior of adding entries to the smallest backend
# dataset first.

dsconfig create-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name example_com_entry_count
--type entry-counter
--set enabled:true
--set "poll-interval:1 m"

# Note that once the entry-count placement algorithm is created
# and enabled, we can delete the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then delete the
# original round-robin algorithm created during the setup
# procedure.

dsconfig delete-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin
```

2. Use `dsconfig` with the --batch-file option to read and execute the commands.

Topology Configuration

Topology configuration enables grouping servers and mirroring configuration changes automatically. It uses a master/slave architecture for mirroring shared data across the topology. All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group. Servers can be added to an existing topology at installation.

Note: To remove a server from the topology, it must be uninstalled with the uninstall tool.

Topology Master Requirements and Selection

A topology master server receives any configuration change from other servers in the topology, verifies the change, then makes the change available to all connected servers. The master always sends a digest of its subtree contents on each update. If the node has a different digest than the master, it knows it's not synchronized. The servers will pull the entire subtree from the master if they detect that they are not synchronized. A server may detect it is not synchronized with the master under the following conditions:

- At the end of its periodic polling interval, if a server's subtree digest differs from that of its master, then it knows it's not synchronized.
- If one or more servers have been added to or removed from the topology, the servers will not be synchronized.

The master of the topology is selected by prioritizing servers by minimum supported product version, most available, newest server version, earliest start time, and startup UUID (a smaller UUID is preferred).

After determining a master, the topology data is reviewed from all available servers (every five seconds by default) to determine if any new information makes a server better suited to being the master. If a new server can be the master, it will communicate that to the other servers, if no other server has advertised that it should be the master. This ensures that all servers accept the same master at approximately the same time (within a few milliseconds of each other). If there is no better master, the initial master maintains the role.

After the best master has been selected for the given interval, the following conditions are confirmed:

- A majority of servers is reachable from that master. (The master server itself is considered while determining this majority.)
- There is only a single master in the entire topology.

If either of these conditions is not met, the topology is without a master and the peer polling frequency is reduced to 100 milliseconds to find a new master as quickly as possible. If there is no master in the topology for more than one minute, a `mirrored-subtree-manager-no-master- found` alarm is raised. If one of the servers in the topology is forced as master with the `force-as-master-for-mirrored-data` option in the Global Configuration object, a `mirrored-subtree-manager-forced-as-master-warning` warning alarm is raised. If multiple servers have been forced as masters, then a `mirrored-subtree-manager-forced-as-master-error` critical alarm will be raised.

Topology Components

When a server is installed, it can be added to an existing topology, which will clone the server's configuration. Topology settings are designed to operate without additional configuration. If required, some settings can be adjusted to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the Global Configuration and in the Config File Handler Backend. Though they are topology settings, they are unique to each server and are not mirrored. Settings must be kept the same on all servers.

The Global Configuration object contains a single topology setting, force-as-master-formirrored- data. This should be set to true on only one of the servers in the topology, and is used only if a situation occurs where the topology cannot determine a master because a majority of servers is not available. A server with this setting enabled will be assigned the role of master, if no suitable master can be determined.

The Config File Handler Backend defines three topology (`mirrored-subtree`) settings:

- `mirrored-subtree-peer-polling-interval` – Specifies the frequency at which the server polls its topology peers to determine if there are any changes that may warrant a new master selection. A lower value will ensure a faster failover, but it will also cause more traffic among the peers. The default value is five seconds. If no suitable master is found, the polling frequency is adjusted to 100 milliseconds until a new master is selected.
- `mirrored-subtree-entry-update-timeout` – Specifies the maximum length of time to wait for an update operation (add, delete, modify or modify-dn) on an entry to be applied by the master on all of the servers in the topology. The default is 10 seconds. In reality, updates can take up to twice as much time as this timeout value if master selection is in progress at the time the update operation was received.
- `mirrored-subtree-search-timeout` – Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

Topology meta-data is stored under the `cn=topology,cn=config` subtree and cluster data is stored under the `cn=cluster,cn=config` subtree. The only setting that can be changed is the cluster name.

Monitor Data for the Topology

Each server has a monitor that exposes that server's view of the topology in its monitor backend, so that peer servers can periodically read this information to determine if there are changes in the topology. Topology data includes the following:

- The server ID of the current master, if the master is not known.
- The instance name of the current master, or if a master is not set, a description stating why a master is not set.
- A flag indicating if this server thinks that it should be the master.
- A flag indicating if this server is the current master.
- A flag indicating if this server was forced as master.
- The total number of configured peers in the topology group.
- The peers connected to this server.
- The current availability of this server.
- A flag indicating whether or not this server is not synchronized with its master, or another node in the topology if the master is unknown.
- The amount of time in milliseconds where multiple masters were detected by this server.
- The amount of time in milliseconds where no suitable server is found to act as master.
- A SHA-256 digest encoded as a base-64 string for the current subtree contents.

The following metrics are included if this server has processed any operations as master:

- The number of operations processed by this server as master.
- The number of operations processed by this server as master that were successful.
- The number of operations processed by this server as master that failed to validate.
- The number of operations processed by this server as master that failed to apply.
- The average amount of time taken (in milliseconds) by this server to process operations as the master.
- The maximum amount of time taken (in milliseconds) by this server to process an operation as the master.

Updating the Server Instance Listener Certificate

To change the SSL certificate for the server, update the keystore and truststore files with the new certificate. The certificate file must have the new certificate in PEM-encoded format, such as:

```
-----BEGIN CERTIFICATE-----

 MIIDKTCCAhGgAwIBAgIEacgGrDANBgkqhkiG9w0BAQsFADBFMR4wHAYDVQQKExVVbmJvdW5kSUQgQ2VydGlmaWN

 GUxIzAhBgNVBAMTGnZtLW1lZGl1bS03My51bmJvdW5kaWQubGFiMB4XDTE1MTAxMjE1MzU0OFoXDTM1MTAwNzE1

 U0OFowRTEeMBwGA1UEChMVVW5ib3VuZElEIENlcnRpZmljYXRlMSMwIQYDVQQDExp2bS1tZWRpdW0tNzMudW5ib

 uZGlkLmxhYjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKN4tAN3o9Yw6Cr9hivwVDxJqF6+aEi9Ir
            GFYLSrggRNXsiAOfWkSMWdIC5vyF5OJ9DlIgvHL4OuqP/
YNEGzKDkgr6MwtUeVSK14+dCixygJGC0nY7k+f0WSCjt
            IHzrmc4WWdrZXmgb
+qv9LupS30JG0FXtcbGkYpjaKXIEqMg4ekz3B5cAvE0SQUFyXEdN4rWOn96nVFkb2CstbiPzA

 gne2tu7paJ6SGFOW0UF7v018XY1m2WHBIoD0WC8nOVLTG9zFUavaOxtlt1TlhClkI4HRMNg8n2EtSTdQRizKuw9
            TXJBb6Kfvnp/
nI73VHRyt47wUVueehEDfLtDP8pMCAwEAAaMhMB8wHQYDVR0OBBYEFMrwjWxl2K+yd9+Y65oKn0g5
            jITgMA0GCSqGSIb3DQEBCwUAA4IBAQBpsBYodblUGew+HewqtO2i8Wt+vAbt31zM5/
kRvo6/+iPEASTvZdCzIBcgl

 etxKGKeCQ0GPeHr42+erakiwmGDlUTYrU3LU5pTGTDLuR2IllTT5xlEhCWJGWipW4q3Pl3cX/9m2ffY/
JLYDfTJao

 JvnXrh7Sg719skkHjWZQgOHXlkPLx5TxFGhAovE1D4qLVRWGohdpWDrIgFh0DVfoyAn1Ws9ICCXdRayajFI4Lc6
            m6SA5+25Y9nno8BhVPf4q5OW6+UDc8MsLbBsxpwvR6RJ5cv3ypfOriTehJsG
+9ZDo7YeqVsTVGwAlW3PiSd9bYP/8
            yu9Cy+0MfcWcSeAE
            -----END CERTIFICATE-----
```

If clients that already have a secure connection established with this server need to be maintained, information about both certificates can reside in the same file (each with their own begin and end headers and footers).

After the keystore and truststore files are updated, run the following dsconfig command to update the server's certificate in the topology registry:

```
$ bin/dsconfig set-server-instance-listener-prop \
  --instance-name server-instance-name \
  --listener-name ldap-listener-mirrored-config \
  --set listener-certificate path-to-new-certificate-file
```

The `listener-certificate` in the topology registry is like a trust store. The public certificates that it has are automatically trusted by the local server. When the local server attempts a secure LDAP connection to a peer, and the peer presents it with its certificate, the local server will check the `listener-certificate` property for that server in the topology registry. If the property contains the peer server's certificate, the local server will trust the peer.

Remove the Self-signed Certificate

The server is installed with a self-signed certificate and key (`ads-certificate`), which are used for internal purposes such as replication authentication, inter-server authentication in the topology registry, reversible password encryption, and encrypted backup or LDIF export. The ads-certificate lives in the keystore file called `ads-truststore` under the server’s `/config` directory. If your deployment requires removing the self-signed certificate, it can be replaced.

The certificate is stored in the topology registry, which enables replacing it on one server and having it mirrored to all other servers in the topology. Any change is automatically mirrored on other servers in the topology. It is stored in human-readable PEM-encoded format and can be updated with dsconfig. The following general steps are required to replace the self-signed certificate:

1. Prepare a new keystore with the replacement key-pair.
2. Update the server configuration to use the new certificate by adding it to the server's list of certificates in the topology registry so that it is trusted by other servers.
3. Update the server's `ads-truststore` file to use the new key-pair.
4. Retire the old certificate by removing it from the topology registry.

Note: Replacing the entire key-pair instead of just the certificate associated with the original private key can make existing backups and LDIF exports invalid. This should be performed immediately after setup or before the key-pair is used. After the first time, only the certificate associated with the private key should have to be changed, for example, to extend its validity period or replace it with a certificate signed by a different CA.

Prepare a New Keystore with the Replacement Key-pair

The self-signed certificate can be replaced with an existing key-pair, or the certificate associated with the original key-pair can be used.

To Use an Existing Key-pair

If a private key and certificate(s) in PEM-encoded format already exist, both the original private key and self-signed certificate can be replaced in ads-truststore with the `manage-certificates` tool.

- The following command imports the keystore file, ads-truststore.new.

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file existing.key \
  --certificate-file existing.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

The certificates listed using the `--certificate-file` options must be ordered so that each subsequent certificate is the issuer for the previous one. So the server certificate comes first, the intermediate certificates next (if any), and the root CA certificate last.

To Use the Certificate Associated with the Original Key-pair

The certificate associated with the original server-generated private key can be replaced with the following commands:

1. Create a CSR for the `ads-certificate`:

```
$ bin/manage-certificates generate-certificate-signing-request \
  --keystore ads-truststore \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --use-existing-key-pair \
  --subject-dn "CN=ldap.example.com,O=Example Corporation,C=US" \
  --output-file ads.csr
```

2. Submit `ads.csr` to a CA for signing.
3. Export the server's private key into `ads.key`:

```
$ bin/manage-certificates export-private-key \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --output-file ads.key
```

4. Import the certificates obtained from the CA (the CA-signed server certificate, any intermediate certificates, and root CA certificate) into `ads-truststore.new`:

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file ads.key \
  --certificate-file new-ads.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

To Update the Server Configuration to Use the New Certificate

To update the server to use the desired key-pair, the `inter-server-certificate` property for the server instance must first be updated in the topology registry. The old and the new certificates may appear within their own begin and end headers in the `inter-server-certificate` property to support transitioning from the old certificate to the new one.

1. Export the server's old ads-certificate into `old-ads.crt`::

```
$ bin/manage-certificates export-certificate \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --export-certificate-chain \
  --output-file old-ads.crt
```

2. Concatenate the old, new certificate, and issuer certificates into one file. On Windows, an editor like notepad can be used. On Unix platforms, use the following command:

```
$ cat old-ads.crt new-ads.crt intermediate.crt root-ca.crt > chain.crt
```

3. Update the `inter-server-certificate` property for the server instance in the topology registry using `dsconfig`:

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name instance-name \
  --set “inter-server-certificate<chain.crt”
```

To Update the ads-truststore File to Use the New Key-pair

The server will still use the old `ads-certificate`. When the new `ads-certificate` needs to go into effect, the old `ads-truststore` file must be replaced with `ads-truststore.new` in the server's `config` directory.

- Move the file.

```
$ mv ads-truststore.new ads-truststore
```

To Retire the Old Certificate

The old certificate is retired by removing it from the topology registry when it has expired. All existing encrypted backups and LDIF exports are not affected because the public key in the old and new server certificates are the same, and the private key will be able to decrypt them.

- Perform the following commands:

```
$ cat new-ads.crt intermediate.crt root-ca.crt<chain.crt
```

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name instance-name \
  --set “inter-server-certificate<chain.crt”
```

Using the Configuration API

PingDirectoryProxy Server provides a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers should allow the application/json content type.

The server includes a servlet extension that provides read and write access to the server's configuration over HTTP. The extension is enabled by default for new installations, and can be enabled for existing deployments by simply adding the extension to one of the server's HTTP Connection Handlers, as follows:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add http-servlet-extension:Configuration
```

The API is made available on the HTTPS Connection handler's host:port in the `/config` context. Due to the potentially sensitive nature of the server's configuration, the HTTPS Connection Handler should be used for hosting the Configuration extension.

Authentication and Authorization with the Configuration API

Clients must use HTTP Basic authentication to authenticate to the Configuration API. If the username value is not a DN, then it will be resolved to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper, or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. For example:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Configuration \
  --set "identity-mapper:Alternative Identity Mapper”
```

To access configuration information, users must have the appropriate privileges:

- To access the `cn=config` backend, users must have the `bypass-acl` privilege or be allowed access to the configuration using an ACI.
- To read configuration information, users must have the `config-read` privilege.
- To update the configuration, users must have the `config-write` privilege.

Relationship Between the Configuration API and the dsconfig Tool

The Configuration API is designed to mirror the `dsconfig` tool, using the same names for properties and object types. Property names are presented as hyphen case in `dsconfig` and as camel-case attributes in the API. In API requests that specify property names, case is not important. Therefore, `baseDN` is the same as `baseDn`. Object types are represented in hyphen case. API paths mirror what is in `dsconfig`. For example, the `dsconfig list-connection-handlers` command is analogous to the API's `/config/connection-handlers` path. Object types that appear in the schema URNs adhere to a `type:subtype` syntax. For example, a Local DB Backend's schema URN is `urn:unboundid:schemas:configuration:2.0:backend:local-db`. Like the `dsconfig` tool, all configuration updates made through the API are recorded in `logs/config-audit.log`.

The API includes the filter, sort, and pagination query parameters described by the SCIM specification. Specific attributes may be requested using the attributes query parameter, whose value must be a comma-delimited list of properties to be returned, for example `attributes=baseDN,description`. Likewise, attributes may be excluded from responses by specifying the `excludedAttributes` parameter.

Operations supported by the API are those typically found in REST APIs:

HTTP Method	Description	Related dsconfig Example
GET	Lists the properties of an object when used with a path representing an object, such as `/config/global-configuration` or `/config/backends/userRoot`. Can also list objects when used with a path representing a parent relation, such as `/config/backends`.	`get-backend-prop`, `list-backends`, `get-global-configuration-prop`
POST	Creates a new instance of an object when used with a relation parent path, such as `/config/backends`.	`create-backend`
PUT	Replaces the existing properties of an object. A PUT operation is similar to a PATCH operation, except that the PATCH identifies the difference between an existing target object and a supplied source object. Only those properties in the source object are modified in the target object. The target object is specified using a path, such as `/config/backends/userRoot`.	`set-backend-prop`, `set-global-configuration-prop`
PATCH	Updates the properties of an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`set-backend-prop`, `set-global-configuration-prop`
DELETE	Deletes an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`delete-backend`

The OPTIONS method can also be used to determine the operations permitted for a particular path.

Object names, such as `userRoot` in the Description column, must be URL-encoded for use in the path segment of a URL. For example, `%20` must be used in place of spaces, and `%25` is used in place of the percent (%) character. The URL for accessing the HTTP Connection Handler object is:

```
/config/connection-handlers/http%20connection%20handler
```

GET Example

The following is a sample GET request for information about the `userRoot` backend:

```
GET /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
```

The response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
   "id": "userRoot",
   "meta": {
     "resourceType": "Local DB Backend",
     "location": "http://localhost:5033/config/backends/userRoot"
   },
   "backendID": "userRoot2",
   "backgroundPrime": "false",
   "backupFilePermissions": "700",
   "baseDN": [
     "dc=example2,dc=com"
   ],
   "checkpointOnCloseCount": "2",
   "cleanerThreadWaitTime": "120000",
   "compressEntries": "false",
   "continuePrimeAfterCacheFull": "false",
   "dbBackgroundSyncInterval": "1 s",
   "dbCachePercent": "10",
   "dbCacheSize": "0 b",
   "dbCheckpointerBytesInterval": "20 mb",
   "dbCheckpointerHighPriority": "false",
   "dbCheckpointerWakeupInterval": "1 m",
   "dbCleanOnExplicitGC": "false",
   "dbCleanerMinUtilization": "75",
   "dbCompactKeyPrefixes": "true",
   "dbDirectory": "db",
   "dbDirectoryPermissions": "700",
   "dbEvictorCriticalPercentage": "0",
   "dbEvictorLruOnly": "false",
   "dbEvictorNodesPerScan": "10",
   "dbFileCacheSize": "1000",
   "dbImportCachePercent": "60",
   "dbLogFileMax": "50 mb",
   "dbLoggingFileHandlerOn": "true",
   "dbLoggingLevel": "CONFIG",
   "dbNumCleanerThreads": "0",
   "dbNumLockTables": "0",
   "dbRunCleaner": "true",
   "dbTxnNoSync": "false",
   "dbTxnWriteNoSync": "true",
   "dbUseThreadLocalHandles": "true",
   "deadlockRetryLimit": "10",
   "defaultCacheMode": "cache-keys-and-values",
   "defaultTxnMaxLockTimeout": "10 s",
   "defaultTxnMinLockTimeout": "10 s",
   "enabled": "false",
   "explodedIndexEntryThreshold": "4000",
   "exportThreadCount": "0",
   "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
   "externalTxnDefaultMaxLockTimeout": "100 ms",
   "externalTxnDefaultMinLockTimeout": "100 ms",
   "externalTxnDefaultRetryAttempts": "2",
   "hashEntries": "false",
   "id2childrenIndexEntryLimit": "66",
   "importTempDirectory": "import-tmp",
   "importThreadCount": "16",
   "indexEntryLimit": "4000",
   "isPrivateBackend": "false",
   "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
   "jeProperty": [
     "je.cleaner.adjustUtilization=false",
     "je.nodeMaxEntries=32"
   ],
   "numRecentChanges": "50000",
   "offlineProcessDatabaseOpenTimeout": "1 h",
   "primeAllIndexes": "true",
   "primeMethod": [
     "none"
   ],
   "primeThreadCount": "2",
   "primeTimeLimit": "0 ms",
   "processFiltersWithUndefinedAttributeTypes": "false",
   "returnUnavailableForUntrustedIndex": "true",
   "returnUnavailableWhenDisabled": "true",
   "setDegradedAlertForUntrustedIndex": "true",
   "setDegradedAlertWhenDisabled": "true",
   "subtreeDeleteBatchSize": "5000",
   "subtreeDeleteSizeLimit": "5000",
   "uncachedId2entryCacheMode": "cache-keys-only",
   "writabilityMode": "enabled"
}
```

GET List Example

The following is a sample GET request for all local backends:

```
GET /config/backends/
Host: example.com:5033
Accept: application/scim+json
```

The response (which has been shortened):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=topology,cn=config"
      ],
      "enabled": "true",
      "isPrivateBackend": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.LDIFBackend",
      "ldifFile": "config/admin-backend.ldif",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "false",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
      ],
      "id": "ads-truststore",
      "meta": {
        "resourceType": "Trust Store Backend",
        "location": "http://localhost:5033/config/backends/ads-
truststore"
      },
      "backendID": "ads-truststore",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=ads-truststore"
      ],
      "enabled": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.TrustStoreBackend",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "true",
      "trustStoreFile": "config/server.keystore",
      "trustStorePin": "********",
      "trustStoreType": "JKS",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:alarm"
      ],
      "id": "alarms",
      "meta": {
        "resourceType": "Alarm Backend",
        "location": "http://localhost:5033/config/backends/alarms"
      },
  ...
```

PATCH Example

Configuration can be modified using the HTTP PATCH method. The PATCH request body is a JSON object formatted according to the SCIM patch request. The Configuration API, supports a subset of possible values for the path attribute, used to indicate the configuration attribute to modify.

The configuration object's attributes can be modified in the following ways. These operations are analogous to the `dsconfig modify-[object]` options.

- An operation to set the single-valued `description` attribute to a new value:

  ```
  {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop
    --backend-name userRoot \
    --set "description:A new backend"
  ```
- An operation to add a new value to the multi-valued `jeProperty` attribute:

  ```
  {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }
  ```

  is analogous to:

  ```
  $ dsconfig  set-backend-prop --backend-name userRoot \
    --add je-property:je.env.backgroundReadLimit=0
  ```
- An operation to remove a value from a multi-valued property. In this case, path specifies a SCIM filter identifying the value to remove:

  ```
  {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }
  ```

  is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --remove je-property:je.cleaner.adjustUtilization=false
```

- A second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify, and a SCIM filter whose attribute is value:

  ```
  {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --remove je-property:je.nodeMaxEntries=32
  ```

- An option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value:

  ```
  {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --reset id2childrenIndexEntryLimit
  ```

The following is the full example request. The API responds with the entire modified configuration object, which may include a SCIM extension attribute `urn:unboundid:schemas:configuration:messages` containing additional instructions:

```
PATCH /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json

{
  "schemas" : [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  } ]
}
```

Example response:

```
{
 "schemas": [
```

```
  "urn:unboundid:schemas:configuration:2.0:backend:local-db"
],
"id": "userRoot2",
"meta": {
  "resourceType": "Local DB Backend",
  "location": "http://example.com:5033/config/backends/userRoot2"
},
"backendID": "userRoot2",
"backgroundPrime": "false",
"backupFilePermissions": "700",
"baseDN": [
  "dc=example2,dc=com"
],
"checkpointOnCloseCount": "2",
"cleanerThreadWaitTime": "120000",
"compressEntries": "false",
"continuePrimeAfterCacheFull": "false",
"dbBackgroundSyncInterval": "1 s",
"dbCachePercent": "10",
"dbCacheSize": "0 b",
"dbCheckpointerBytesInterval": "20 mb",
"dbCheckpointerHighPriority": "false",
"dbCheckpointerWakeupInterval": "1 m",
"dbCleanOnExplicitGC": "false",
"dbCleanerMinUtilization": "75",
"dbCompactKeyPrefixes": "true",
"dbDirectory": "db",
"dbDirectoryPermissions": "700",
"dbEvictorCriticalPercentage": "0",
"dbEvictorLruOnly": "false",
"dbEvictorNodesPerScan": "10",
"dbFileCacheSize": "1000",
"dbImportCachePercent": "60",
"dbLogFileMax": "50 mb",
"dbLoggingFileHandlerOn": "true",
"dbLoggingLevel": "CONFIG",
"dbNumCleanerThreads": "0",
"dbNumLockTables": "0",
"dbRunCleaner": "true",
"dbTxnNoSync": "false",
"dbTxnWriteNoSync": "true",
"dbUseThreadLocalHandles": "true",
"deadlockRetryLimit": "10",
"defaultCacheMode": "cache-keys-and-values",
"defaultTxnMaxLockTimeout": "10 s",
"defaultTxnMinLockTimeout": "10 s",
"description": "123",  "enabled": "false",
"explodedIndexEntryThreshold": "4000",
"exportThreadCount": "0",
"externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
"externalTxnDefaultMaxLockTimeout": "100 ms",
"externalTxnDefaultMinLockTimeout": "100 ms",
"externalTxnDefaultRetryAttempts": "2",
"hashEntries": "false",
"importTempDirectory": "import-tmp",
"importThreadCount": "16",
"indexEntryLimit": "4000",
"isPrivateBackend": "false",
"javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
"jeProperty": [  "\"je.env.backgroundReadLimit=0\""
],
"numRecentChanges": "50000",
"offlineProcessDatabaseOpenTimeout": "1 h",
"primeAllIndexes": "true",
 "primeMethod": [
   "none"
 ],
 "primeThreadCount": "2",
 "primeTimeLimit": "0 ms",
 "processFiltersWithUndefinedAttributeTypes": "false",
 "returnUnavailableForUntrustedIndex": "true",
 "returnUnavailableWhenDisabled": "true",
 "setDegradedAlertForUntrustedIndex": "true",
 "setDegradedAlertWhenDisabled": "true",
 "subtreeDeleteBatchSize": "5000",
 "subtreeDeleteSizeLimit": "5000",
 "uncachedId2entryCacheMode": "cache-keys-only",
 "writabilityMode": "enabled",
 "urn:unboundid:schemas:configuration:messages:2.0": {
   "requiredActions": [
     {
       "property": "jeProperty",
       "type": "componentRestart",
       "synopsis": "In order for this modification to take effect,
          the component must be restarted, either by disabling and
          re-enabling it, or by restarting the server"
     },
     {
       "property": "id2childrenIndexEntryLimit",
       "type": "other",
       "synopsis": "If this limit is increased, then the contents
          of the backend must be exported to LDIF and re-imported to
          allow the new limit to be used for any id2children keys
          that had already hit the previous limit."
     }
   ]
 }
}
```

Configuration API Paths

The Configuration API is available under the /config path. A full listing of supported sub-paths is available by accessing the base `/config/ResourceTypes` endpoint:

```
GET /config/ResourceTypes
Host: example.com:5033
Accept: application/scim+json
```

Sample response (abbreviated):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "dsee-compat-access-control-handler",
      "name": "DSEE Compat Access Control Handler",
      "description": "The DSEE Compat Access Control
              Handler provides an implementation that uses syntax
              compatible with the Sun Java System Directory Server
              Enterprise Edition access control handler.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/dsee-compat-
access-control-handler"
      }
    },
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "access-control-handler",
      "name": "Access Control Handler",
      "description": "Access Control Handlers manage the
              application-wide access control. The server's access
              control handler is defined through an extensible
              interface, so that alternate implementations can be created.
              Only one access control handler may be active in the server
                        at any given time.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/access-
control-handler"
      }
    },
    {
 ...
```

The response's `endpoint` elements enumerate all available sub-paths. The path `/config/access-control-handler` in the example can be used to get a list of existing access control handlers, and create new ones. A path containing an object name such as `/config/backends/{backendName}`, where `{backendName}` corresponds to an existing backend (such as `userRoot`) can be used to obtain an object’s properties, update the properties, or delete the object.

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted nor created. These paths can be differentiated from others by their singular, rather than plural, relation name (for example `global-configuration`).

Sorting and Filtering Objects

The Configuration API supports SCIM parameters for filter, sorting, and pagination. Search operations can specify a SCIM filter used to narrow the number of elements returned. See the SCIM specification for the full set of operations for SCIM filters. Clients can also specify sort parameters, or paging parameters. Include or exclude attributes can be specified in both get and list operations.

GET Parameter	Description
filter	Values can be simple SCIM filters such as `id eq "userRoot"` or compound filters like `meta.resourceType eq "Local DB Backend"` and `baseDn co "dc=exmple,dc=com"`.
sortBy	Specifies a property value by which to sort.
sortOrder	Specifies either `ascending` or `descending` alphabetical order.
startIndex	1-based index of the first result to return.
count	Indicates the number of results per page.

Updating Properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH. With PUT, the server computes the differences between the object in the request with the current version in the server, and performs modifications where necessary. The server will never remove attributes that are not specified in the request. The API responds with the entire modified object.

Request:

```
PUT /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
{
  "description" : "A new description."
}
```

Response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode":
  "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "abc",
  "enabled": "true",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior":
  "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "true",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "numRecentChanges": "50000",  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative Actions

Updating a property may require an administrative action before the change can take effect. If so, the server will return `200 Success`, and any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

For example, changing the `jeProperty` of a backend will result in the following:

```
"urn:unboundid:schemas:configuration:messages:2.0": {
    "required-actions": [
      {
       "property": "baseContextPath",
       "type": ""componentRestart",
       "synopsis": "In order for this modification to take effect, the
 component
                 must be restarted, either by disabling and re-enabling it,
 or
                 by restarting the server"
      },
      {
        "property": {
        "type": "other",
        "synopsis": "If this limit is increased, then the
                 contents of the backend must be exported to LDIF
                 and re-imported to allow the new limit to be used
```

```
                  for any id2children keys that had already hit the
                  previous limit."
      }
    ]
  }
```

Updating Servers and Server Groups

Servers can be configured as part of a server group, so that configuration changes that are applied to a single server, are then applied to all servers in a group. When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query attribute. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `single-server` or `server-group` can be specified. For example:

```
http://localhost:8082/config/backends/userRoot?applyChangeTo=single-server
```

Configuration API Responses

Clients of the API should examine the HTTP response code in order to determine the success or failure of a request. The following are response codes and their meanings:

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, or object properties, administrative actions.
204 No Content	The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation.	None.
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message.
401 Unauthorized	User authentication is required. Some user agents such as browsers may respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid.	None.
403 Forbidden	The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted.	None.
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message.
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists, or an attempt was made to delete an object that is referred to by another object.	Error summary and optional message.

Response Code	Description	Response Body
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None.
500 Server Error	The server encountered an unexpected error. Please report server errors to customer support.	Error summary and optional message.

An application that uses the Configuration API should limit dependencies on particular text appearing in error message content. These messages may change, and their presence may depend on server configuration. Use the HTTP return code and the context of the request to create a client error message. The following is an example encoded error message:

```
{
 "schemas": [
   "urn:ietf:params:scim:api:messages:2.0:Error"
 ],
 "status": 404,
 "scimType": null,
 "detail": "The Local DB Index does not exist."
}
```

Generating a Summary of Configuration Components

The Directory Proxy Server provides a `config-diff` tool that generates a summary of the configuration in a local or remote directory server instance. The tool is useful when comparing configuration settings on the directory server instance when troubleshooting issues or when verifying configuration settings on newly-added servers to your network. The tool can interact with the local configuration regardless of whether the server is running or not.

Run the `config-diff` --help option to view other available tool options.

To Generate a Summary of Configuration Components

- Run the `config-diff` tool to generate a summary of the configuration components on the directory server instance. The following command runs a summary on a local online server.

  ```
  $ bin/config-diff
  ```
- The following example compares the current configuration of the local server to the baseline, pre-installation configuration, ignoring any changes that could be made by the installer, and writes the output to the `configuration-steps.dsconfig` file. This provides a script that can be used to configure a newly installed server identically to the local server:

  ```
  $ bin/config-diff --sourceLocal \
    --sourceBaseline \
    --targetLocal \
    --exclude differs-after-install \
    --outputFile configuration-steps.dsconfig
  ```

Configuring Server Groups

The PingDirectoryProxy Server provides a mechanism for setting up administrative domains that synchronize configuration changes among servers in a server group. After you have set up a server group, you can make an update on one server using `dsconfig`, then you can apply the change to the other servers in the group using the --applyChangeTo server-group option of the `dsconfig` non-interactive command. If you want to apply the change to one server in the group, use the --applyChangeTo single-server option. When using `dsconfig` in interactive command-line mode, you will be asked if you want to apply the change to a single server or to all servers in the server group.

About the Server Group Example

You can create an administrative server group using the `dsconfig` tool. The general process is to create a group, add servers to the group, and then set a global configuration property to use the server group. If you are configuring a replication topology, then you must configure the replicas to be in a server group as outlined in Replication Configuration.

The following example procedure adds three Directory Proxy Server instances into the server group labelled "group-one".

To Create a Server Group

1. Create a group called “group-one” using `dsconfig`.

   ```
   $ bin/dsconfig create-server-group --group-name group-one
   ```

2. Add any directory server to the server group. If you have set up replication between a set of servers, these server entries will have already been created by the `dsreplication` enable command.

   ```
   $ bin/dsconfig set-server-group-prop --group-name group-one --set
    member:server1

   $ bin/dsconfig set-server-group-prop --group-name group-one --set
    member:server2

   $ bin/dsconfig set-server-group-prop --group-name group-one --set
    member:server3
   ```

3. Set a global configuration property for each of the servers that should share changes in this group.

   ```
   $ bin/dsconfig set-global-configuration-prop \
     --set configuration-server-group:group-one
   ```

4. Test the server group. In this example, enable the log publisher for each directory server in the group, server-group, by using the --applyChangeTo server-group option.

   ```
   $ bin/dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Audit Logger" \
     --set enabled:true \
     --applyChangeTo server-group
   ```

5. View the property on the first directory server instance.

   ```
   $ bin/dsconfig get-log-publisher-prop \
     --publisher-name "File-Based Audit Logger" \
     --property enabled
   ```

   ```
   Property : Value(s)
   ---------:---------
   enabled : true
   ```

6. Repeat step 5 on the second and third directory server instance.

7. Test the server group by disabling the log publisher on the first directory server instance by using the --applyChangeTo single-server.

   ```
   $ bin/dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Audit Logger" \
     --set enabled:disabled \
     --applyChangeTo single-server
   ```

8. View the property on the first directory server instance. The first directory server instance should be disabled.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : false
```

9. View the property on the second directory server instance. Repeat this step on the third directory server instance to verify that the property is still enabled on that server.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : true
```

Domain Name Service (DNS) Caching

If needed, two global configuration properties can be used to control the caching of hostname-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system. Use the `dsconfig` tool to configure these properties.

- **network-address-cache-ttl** – Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a hostname-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.
- **network-address-outage-cache-enabled** – Caches hostname-to-IP address results in the event of a DNS outage. This is set to true by default, meaning name resolution results are cached. Unexpected service interruptions may occur during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

IP Address Reverse Name Lookups

PingDirectoryProxy Server does not explicitly perform numeric IP address-to-hostname lookups. However, address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing may trigger implicit reverse name lookups. For more information about how address masks are configured in the server, review the following information for each server:

- ACI dns: bind rules under *Managing Access Control* (Directory Server and Directory Proxy Server)
- `ds-auth-allowed-address`: *Adding Operational Attributes that Restrict Authentication* (Directory Server)
- Connection Criteria: *Restricting Server Access Based on Client IP Address* (Directory Server and Directory Proxy Server)
- Connection Handlers: restrict server access using Connection Handlers (Configuration Reference Guide for all servers)

Configuring Traffic Through a Load Balancer

If a PingDirectoryProxy Server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, then it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request. If the actual client's IP address must be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and PingDirectoryProxy Server. For PingDirectoryProxy Servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP), and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

On the load balancer, configure settings to provide the `X-Forwarded-*` information, such as `X-Forwarded-Host:`. See the product documentation for the device type.

Managing Root Users Accounts

The PingDirectoryProxy Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file (for example, under `cn=Root DNs,cn=config`). The root user is the LDAP-equivalent of a UNIX super-user account and inherits its read-write privileges from the default root privilege set. Root users can be created and updated with the `dsconfig` tool. Root user entries are stored in the server’s configuration.The following is a sample command to create a new root user:

```
bin/dsconfig create-root-dn-user --user-name "Joanne Smith" \
  --set last-name:Smith \
  --set first-name:Joanne \
  --set user-id:jsmith \
  --set 'email-address:jsmith@example.com' \
  --set mobile-telephone-number:8889997777 \
  --set home-telephone-number:5556667777 \
  --set work-telephone-number:4445556666
```

To limit full access to all of the Directory Proxy Server, create separate administrator accounts with limited privileges so that you can identify the administrator responsible for a particular change. Having separate user accounts for each administrator also makes it possible to enable password policy functionality (such as password expiration, password history, and requiring secure authentication) for each administrator.

Default Root Privileges

The PingDirectoryProxy Server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments.

Note: Creating restricted root user accounts requires assigning privileges and necessary access controls for actions on specific data or backends. Access controls are determined by how the directory is configured and the structure of your data. See Chapter 16: Managing Access Controls for more information.

The following set of root privileges are available to each root user DN:

Table 2: Default Root Privileges

Privilege	Description
audit-data-security	Allows the associated user to execute data security auditing tasks.
backend-backup	Allows the user to perform backend backup operations.
backend-restore	Allows the user to perform backend restore operations.
bypass-acl	Allows the user to bypass access control evaluation.
config-read	Allows the user to read the server configuration.

Privilege	Description
config-write	Allows the user to update the server configuration.
disconnect-client	Allows the user to terminate arbitrary client connections.
ldif-export	Allows the user to perform LDIF export operations.
ldif-import	Allows the user to perform LDIF import operations.
lockdown-mode	Allows the user to request a server lockdown.
manage-topology	Allows the user to modify topology setting.
metrics-read	Allows the user to read server metrics.
modify-acl	Allows the user to modify access control rules.
password-reset	Allows the user to reset user passwords but not their own. The user must also have privileges granted by access control to write the user password to the target entry.
permit-get-password-policy-state-issues	Allows the user to access password policy state issues.
privilege-change	Allows the user to change the set of privileges for a specific user, or to change the set of privileges automatically assigned to a root user.
server-restart	Allows the user to request a server restart.
server-shutdown	Allows the user to request a server shutdown.
soft-delete-read	Allows the user access to soft-deleted entries.
stream-values	Allows the user to perform a stream values extended operation that obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
third-party-task	Allows the associated user to invoke tasks created by third-party developers.
unindexed-search	Allows the user to perform an unindexed search in the Oracle Berkeley DB Java Edition backend.
update-schema	Allows the user to update the server schema.
use-admin-session	Allows the associated user to use an administrative session to request that operations be processed using a dedicated pool of worker threads.

The Directory Proxy Server provides other privileges that are not assigned to the root user DN by default but can be added using the `ldapmodify` tool (see Modifying Individual Root User Privileges) for more information.

Table 3: Other Available Privileges

Privilege	Description
bypass-pw-policy	Allows the associated user bypass password policy rules and restrictions.
bypass-read-aci	Allows the associated user to bypass access control checks performed by the server for bind, compare, and search operations. Access control evaluation may still be enforced for other types of operations.
jmx-notify	Allows the associated user to subscribe to receive JMX notifications.
jmx-read	Allows the associated user to perform JMX read operations.
jmx-write	Allows the associated user to perform JMX write operations.

Privilege	Description
permit-externally-processed-authentication	Allows the associated user accept externally processed authentication.
permit-proxied-mschapv2-details	Allows the associated user to permit MS-CHAP V2 handshake protocol.
proxied-auth	Allows the associated user to accept proxied authorization.

Configuring Locations

PingDirectoryProxy Server defines locations, both for the LDAP external servers and the proxy server instances themselves. A location defines a collection of servers that share access and latency characteristics. For example, your deployment might include two data centers, one in the east and one in the west. These data centers would be configured as two locations in the Directory Proxy Server. Each location is associated with a name and an ordered list of failover locations, which could be used if none of the servers in the preferred location are available. You can define these locations using the Administrative Console or the command line.

The Directory Proxy Server itself is also associated with a location. This location is specified in the global configuration properties of the Directory Proxy Server. If the load balancing algorithm's `use-location` property is set to true, then the load balancing component of the Directory Proxy Server refers to the Directory Proxy Server's location to determine the external servers it prefers to communicate with.

To Configure Locations Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server main menu, enter the number corresponding to location configuration. Then, enter the number to create a new location.
7. Enter the name of the new location. This example demonstrates configuring a location called East. Enter **f** to finish configuring the location. Repeat this procedure to create a location called West.

```
>>>>  Enter a name for the location that you want to create: east

>>>>  Configure the properties of the location

      Property                   Value(s)
      -----------------------------------
```

```
   1) description                  -
   2) preferred-failover-location  -

   ?) help
   f) finish - create the new location
   d) display the equivalent dsconfig arguments to
      create this object
   b) back
   q) quit

Enter choice [b]: f
```

8. Next, edit the configuration of an existing location, in this example a location named East.

```
>>>> Location menu
What would you like to do?

  1) List existing locations
  2) Create a new location
  3) View and edit an existing location
  4) Delete an existing location

  b) back
  q) quit

Enter choice [b]: 3

>>>> Select the location from the following list:
   1) East
   2) West

   b) back
   q) quit

Enter choice [b]: 1
```

9. Define the preferred failover location property for East. This property provides alternate locations that can be used if servers in this location are not available. If more than one location is provided, the Directory Proxy Server tries the locations in the order listed.

```
 >>>> Configure the properties of the Location

     Property                   Value(s)
     -----------------------------------
  1) description                  -
  2) preferred-failover-location  -

  ?) help
  f) finish - create the new location
  d) display the equivalent dsconfig arguments to create this object
  b) back
  q) quit

Enter choice [b]: 2

...

Do you want to modify the 'preferred-failover-location' property?

   1) Add one or more values

   ?) help
   q) quit
```

```
Enter choice [1]: 2

Select the locations you wish to add:

  1) East
  2) West
  3) Create a new location
  4) Add all locations

...

Enter one or more choices separated by commas[b]: 2
```

10. Verify and apply your change to the property.

```
Do you want to modify the 'preferred-failover-location' property?

  1) Use the value: West
  2) Add one or more values
  3) Remove one or more values
  4) Leave undefined
  5) Revert changes

  ?) help
  q) quit

Enter choice [1]:

>>>> Configure the properties of the location

     Property                   Value(s)
     -----------------------------------
  1) description                  -
  2) preferred-failover-location  West

  ?) help
  f) finish - apply any changes to the Location
  d) display the equivalent dsconfig command lines to either
     re-create this object or only to apply pending changes
  b) back
  q) quit

Enter choice [b]: f
```

11. Repeat steps 8 and 9 for the West location, assigning it a failover location of East.

To Modify Locations Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server main menu, enter the number corresponding to Global Configuration. Then enter the number to view and edit the Global Configuration.
7. Enter the number associated with the location configuration property.

```
Enter choice [b]: 2
```

8. Specify a new location for this Directory Proxy Server instance, in this example the East location. Operations involving communications with other servers may prefer servers in the same location to ensure low-latency responses.

```
>>>> Configuring the 'location' property

...

Do you want to modify the 'location' property?

  1) Leave undefined
  2) Change it to the location: East
  3) Change it to the location: West
  4) Create a new location

  b) back
  q) quit

Enter choice [b]: 2
```

9. Enter **f** to finish the operation.

```
Enter choice [b]: f
```

Configuring Batched Transactions

You can configure the Directory Proxy Server to use batched transactions in both simple and entry-balanced configurations. The batched transactions feature supports two implementations: the standard LDAP transactions per RFC 5805 and the PingDirectoryProxy Server proprietary implementation, known as the *multi-update extended operation*. Batched transactions can be used through the Directory Proxy Server in both simple and entry-balanced configurations although only in cases in which all operations within the transaction request may be processed within the same backend server and within the same Berkeley DB JE backend. Batched transactions cannot be processed across multiple servers or multiple Directory Server backends.

The multi-update extended operation makes it possible to submit multiple updates in a single request. These updates may be processed either as individual operations or as a single atomic unit. When the Directory Proxy Server receives a Start Batched Transaction request, it will queue all associated operations in memory until the End Batched Transaction request is received with the intention to commit, at which point the set of operations is sent as a single multi-update extended request to the Directory Server.

Add, delete, modify, modify DN, and password modify extended operations may be included in the set of operations processed during a batch transaction. The operations are processed sequentially in the order in which they were included in the extended request. If an error occurs while processing an operation in the set, then the server can be instructed to continue the processing or to cancel any remaining operations. If the operations are not cancelled, you can configure the server to process all operations as a single atomic unit.

Because of this use of multi-update, the external Directory Server must be configured to allow multi-update extended requests made by the Directory Proxy Server on behalf of the DN submitting the batched transaction. For example, the following Directory Server `dsconfig` command grants anonymous access to the multi-update extended request. The submitter of the request still needs access rights for the individual operations within the multiple-update.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(extop="1.3.6.1.4.1.30221.2.6.17")(version 3.0;
 acl "Anonymous access to multi-update extended request"; allow (read)
 userdn="ldap:///anyone";)'
```

To Configure Batched Transactions

Batched transactions are managed by the Batched Transactions Extended Operation Handler. You can use it to configure the start transaction and end transaction operations used to indicate the set of add, delete, modify, modify DN, and/or password modify operations as a single atomic unit.

1. You can configure batched transactions using the `dsconfig` command as follows:

   ```
   $ bin/dsconfig set-extended-operation-handler-prop \
     --handler-name "Batched Transactions" \
     --set enabled:true
   ```
2. Configure the external servers to allow the multi-update extended operation by granting access rights to the feature. See example in the previous section.

Configuring Server Health Checks

You can use the PingDirectoryProxy Server to configure different types of health checks for your deployment. The health checks define external server availability as either being available, unavailable, or degraded. The external server health is given a value from 0 to 10, which is used to determine if the server is available and how that server compares to other servers with the same state. Load-balancing algorithms can be used to check the score and prefer servers with higher scores over those with lower scores.

An individual health check can be defined for use against all external servers or assigned to individual external servers, as determined by the `use-for-all-servers` parameter within the health check configuration object. If `use-for-all-servers` is set to true, the Directory Proxy Server applies the health check to all external servers in all locations. If `use-for-all-servers` is set to false, then the health check is only employed against an external server if the configuration object for that external server lists the health check.

For more information about health checks and the type of health checks supported by PingDirectoryProxy Server, see *About LDAP Health Checks*.

About the Default Health Checks

By default, the Directory Proxy Server has two health check instances enabled for use on all servers:

- **Consume Admin Alerts**. This health check detects administrative alerts from the Directory Server, as soon as they are issued, by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the Directory Server. When the Directory Proxy Server is notified by the Directory Server of a new alert, it immediately retrieves the base `cn=monitor` entry of the Directory Server. If this entry has a value for the `unavailable-alert-type` attribute, then the Directory Proxy Server will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the Directory Proxy Server will consider it degraded.
- **Get Root DSE**. This health check detects if the root DSE entry exists on the LDAP external server. As this entry always exists on a PingDirectory Server, the absence of the entry suggests that the LDAP external server may be degraded or unavailable.

About Creating a Custom Health Check

You can create a new health check from scratch or use an existing health check as a template for the configuration of a new health check. If you choose to create a custom health check, you can create one of the following types:

- **Admin Alert Health Check**. This health check watches for administrative alerts generated by the LDAP external server to determine whether the server has entered a degraded or unavailable state.
- **Groovy Scripted LDAP Health Check**. This health check allows you to create custom LDAP health checks in a dynamically-loaded Groovy script, which implements the `ScriptedLDAPHealthCheck` class defined in the Server SDK.
- **Replication Backlog Health Check**. While the Admin Alert Health Check consumes replication backlog alerts emitted from external servers, a finer definition of external server health based on replication backlog can be defined with this health check. If a server falls too far behind in replication, then the Directory Proxy Server can stop sending requests to it. A server is classified as degraded or unavailable if the threshold is reached for the number of backlogged changes, the age of the oldest backlogged change, or both.
- **Search LDAP Health Check**. This health check performs searches on an LDAP external server and gauges the health of the server depending if the expected results were returned within an acceptable response time. For example, if an error occurs while attempting to communicate with the server, then the server is considered unavailable. You can also apply filters to the results to use values within the monitor entry as indicators of server health.
- **Third Party LDAP Health Check**. This health check allows you to define LDAP health check implementations in third-party code using the Server SDK.
- **Work Queue Busyness Health Check**. This health check may be used to monitor the percentage of time that worker threads in backend servers spend processing requests.

To Configure a Health Check Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server main menu, enter the number corresponding to LDAP health checks. Enter the number to create a new LDAP Health Check, then press `n` to create a new health check from scratch.
7. Select the type of health check you want to create. This example demonstrates the creation of a new search LDAP health check.

```
>>> Select the type of LDAP Health Check that you want to create:

      1)  Admin Alert LDAP Health Check
      2)  Custom LDAP Health Check
      3)  Groovy Scripted LDAP Health Check
      4)  Replication Backlog LDAP Health Check
      5)  Search LDAP Health Check
      6)  Third Party LDAP Health Check
      7)  Work Queue Busyness LDAP Health Check

      ?) help
      c) cancel
      q) quit

Enter choice [c]: 5
```

8. Specify a name for the new health check. In this example, the health check is named `Get example.com`.

```
>>>> Enter a name for the search LDAP Health Check that you want to create:
 Get example.com
```

9. Enable the new health check.

```
>>>> Configuring the 'enabled' property

Indicates whether this LDAP health check is enabled for use in the server.

Select a value for the 'enabled' property:

   1) true
   2) false

   ?) help
   c) cancel
   q) quit

Enter choice [c]: 1
```

10. Next, configure the properties of the health check. You may need to modify the `base-dn` property, as well as one or more response time thresholds for non-local external servers, accommodating WAN latency. Below is a Search LDAP Health Check for the single entry `dc=example,dc=com`, which allows non-local responses of up to 2 seconds to still be considered healthy.

```
>>>> Configure the properties of the Search LDAP Health Check

           Property                                   Value(s)
           -----------------------------------------------------------
      1)   description                                -
      2)   enabled                                    true
      3)   use-for-all-servers                        false
      4)   base-dn                                    "dc=example,dc=com"
      5)   scope                                      base-object
      6)   filter                                     (objectClass=*)
      7)   maximum-local-available-response-time      1 s
      8)   maximum-nonlocal-available-response-time   2 s
      9)   minimum-local-degraded-response-time       500 ms
      10)  minimum-nonlocal-degraded-response-time    1 s
      11)  maximum-local-degraded-response-time       10 s
      12)  maximum-nonlocal-degraded-response-time    10 s
      13)  minimum-local-unavailable-response-time    5 s
      14)  minimum-nonlocal-unavailable-response-time 5 s
      15)  allow-no-entries-returned                  true
      16)  allow-multiple-entries-returned            true
      17)  available-filter                           -
      18)  degraded-filter                            -
      19)  unavailable-filter                         -

      ?) help
      f) finish - create the new Search LDAP Health Check
      d) display the equivalent dsconfig arguments to create this object
      b) back
      q) quit
```

Configuring LDAP External Servers

The LDAP external server configuration element defines the connection, location, and health check information necessary for the Directory Proxy Server to communicate with the server properly.

PingDirectoryProxy Server includes a tool, `prepare-external-server`, for configuring communication between the Directory Proxy Server and the LDAP backend server. After you add a new LDAP external server to an existing installation, we strongly recommend that you run this tool to automatically create the user account necessary for communications. The `prepare-external-server` tool does not make configuration changes to the local Directory Proxy Server, only the external server is modified. When you run this tool, you must supply the user account and password that you specified for the Directory Proxy Server during configuration, `cn=Proxy User` by default.

Important: You should not use `cn=Directory Manager` as the account to use for communication between the Directory Proxy Server and the Directory Server. For security reasons, the account used to communicate between the Directory Proxy Server and the Directory Server should not be directly accessible by clients accessing the Directory Proxy Server. The account that you choose should meet the following criteria:

- For all server types, it should not exist in the Directory Proxy Server but only in the backend directory server instances.
- For Ping Identity Directory Server, this user should be a root user.
- For Ping Identity Directory Server, this user should not automatically inherit the default set of root privileges, but instead should have exactly the following set of privileges: `bypass-read-acl`, `config-read`, `lockdown-mode`, `proxied-auth`, and stream-values.
- For Sun Directory Servers, the account should be created below the `cn=Root DNs,cn=config` entry and the `nsSizeLimit`, `nsTimeLimit`, `nsLookThroughLimit`, and `nsIdleTimeout` values for the account should be set to -1. You also need to create access control rules to grant the user account appropriate permissions within the server. The `prepare-external-server` tool handles all of this work automatically.

About the prepare-external-server Tool

Use the `prepare-external-server` tool if you have added LDAP external servers using `dsconfig`. The `create-initial-proxy-config` tool automatically runs the `prepare-external-server` tool to configure server communications so that you do not need to invoke it separately. The `create-initial-proxy-config` tool verifies that the proxy user account exists and has the correct password and required privileges. If it detects any problems, it prompts for manager credentials to rectify them.

If you want the `prepare-external-server` tool to add the LDAP external server's certificates to the Directory Proxy Server's trust store, you must include the --proxyTrustStorePath option, and either the --proxyTrustStorePassword or the --proxyTrustStorePasswordFile option. The default location of the Directory Proxy Server trust store is `config/truststore`. The pin is encoded in the `config/truststore.pin` file.

For example, run the tool as follows to prepare a PingDirectory Server on the remote host, ds-east-01.example.com, listening on port 1389 for access by the Directory Proxy Server using the default user account `cn=Proxy User`:

```
prepare-external-server --hostname ds-east-01.example.com \
--port 1389 --baseDN dc=example,dc=com --proxyBindPassword secret
```

When the `prepare-external-server` command above is executed, it creates the `cn=Proxy User` Root DN entry as well as an access control rule in the Directory Server to grant the proxy user the proxy access right.

Note: For non-Ping Identity servers, the --baseDN argument is required for the `prepare-external-server` tool. The base DN is used to create the global ACI entries for these servers.

To Configure Server Communication Using the prepare-external-server Tool

The following example illustrates how to run the `prepare-external-server` tool to prepare a Directory Server on the remote host, ds-east-01.example.com, listening on port 1636. The Directory Server is being accessed by a Directory Proxy Server that uses the default user account `cn=Proxy User,cn=Root DNs,cn=config`. Since a password to the truststore is not provided, the truststore defined in the `--proxyTrustStorePath` is referenced in a read-only manner.

- Use the `prepare-external-server` tool to prepare the Directory Server. Follow the prompts to set up the external server.

```
$ ./PingDirectoryProxy/bin/prepare-external-server \
  --baseDN dc=example,dc=com
  --proxyBindPassword password \
  --hostname ds-east-01.example.com \
  --useSSL \
  --port 1636
  --proxyTrustStorePath /full/path/to/trust/store \
  --proxyTrustStorePassword secret
```

```
Testing connection to ds-east-01.example.com:1636 .....

Do you wish to trust the following certificate?

Certificate Subject: CN=ds-east-01.example.com, O=Example Self-Signed
 Certificate
Issuer Subject:      CN=ds-east-01.example.com, O=Example Self-Signed
 Certificate
Validity:            Thu May 21 08:02:30 CDT 2009 to Wed May 16 08:02:30 CDT
 2029

Enter 'y' to trust the certificate or 'n' to reject it.

y

The certificate was added to the local trust store

Done
Testing 'cn=Proxy User' access to ds-east-01.example.com:1636 ..... Failed
 to bind as
'cn=Proxy User'

Would you like to create or modify root user 'cn=Proxy User' so that it is
 available
for this Directory Proxy Server? (yes / no) [yes]:

Enter the DN of an account on ds-east-01.example.com:1636 with which to
 create or
manage the 'cn=Proxy User' account [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':

Created 'cn=Proxy User,cn=Root DNs,cn=config'

Testing 'cn=Proxy User' privileges ..... Done
```

To Configure an External Server Using dsconfig

1. Use the `dsconfig` tool to create and configure external servers. Then, specify the hostname, connection method, port number, and bind DN as described in previous procedures.

```
$ bin/dsconfig
```

2. In the Directory Proxy Server main menu, enter the number corresponding to external servers. Then, enter the number to create a new external server.
3. Select the type of server you want to create. This example creates a new Ping Identity Directory Server.

```

>>>> Select the type of external server that you want to create:

  1)  Ping Identity DS external server
  2)  JDBC external server
  3)  LDAP external server
  4)  Sun DS external server

  ?)  help
  c)  cancel
  q)  quit

Enter choice [c]: 1
```

4. Specify a name for the new external server. In this example, the external server is named east1.

```
>>>> Enter a name for the Ping Identity DS external server that you want
to create: east1
```

5. Configure the host name or IP address of the target LDAP external server.

```
Enter a value for the 'server-host-name' property:east1.example.com
```

6. Next, configure the location property of the new external server.

```
Do you want to modify the 'location' property?

   1)  Leave undefined
   2)  Change it to the location: East
   3)  Change it to the location: West
   4)  Create a new location

   ?)  help
   q)  quit

Enter choice [1]: 2
```

7. Next, define the bind DN and bind password.

```
Do you want to modify the 'bind-dn' property?

   1)  Leave undefined
   2)  Change the value

   ?)  help
   q)  quit

Enter choice [1]: 2

Enter a value for the 'bind-dn' property [continue]: cn=Proxy User,cn=Root
 DNs,cn=config

Enter choice [b]: 6

...

Do you want to modify the 'password' property?

```

```
   1) Leave undefined
   2) Change the value
   ?) help
   q) quit

Enter choice [1]: 2

Enter a value for the 'password' property [continue]:
Confirm the value for the 'password' property:
```

8. Enter `f` to finish the operation.

```
Enter choice [b]: f
```

```
The Ping Identity DS external server was created successfully.
```

Once you have completed adding the server, run the `prepare-external-server` tool to configure communications between the Directory Proxy Server and the Ping Identity Directory Server(s).

To Configure Authentication with a SASL External Certificate

By default, the Directory Proxy Server authenticates to the Directory Server using LDAP simple authentication (with a bind DN and a password). However, the Directory Proxy Server can be configured to use SASL EXTERNAL to authenticate to the Directory Server with a client certificate.

Have Directory Proxy Server instances installed and configured to communicate with the backend Directory Server instances using either SSL or StartTLS. After the servers are configured, perform the following steps to configure SASL EXTERNAL authentication.

1. Create a JKS keystore that includes a public and private key pair for a certificate that the Directory Proxy Server instance(s) will use to authenticate to the Directory Server instance(s). Run the following command in the instance root of one of the Directory Proxy Server instances. When prompted for a keystore password, enter a strong password to protect the certificate. When prompted for the key password, press **ENTER** to use the keystore password to protect the private key:

```
$ keytool -genkeypair \
  -keystore config/proxy-user-keystore \
  -storetype JKS \
  -keyalg RSA \
  -keysize 2048 \
  -alias proxy-user-cert \
  -dname "cn=Proxy User,cn=Root DNs,cn=config" \
  -validity 7300
```

2. Create a `config/proxy-user-keystore.pin` file that contains a single line that is the keystore password provided in the previous step.
3. If there are other Directory Proxy Server instances in the topology, copy the `proxy-user-keystore` and `proxy-user-keystore.pin` files into the `config` directory for all instances.
4. Use the following command to export the public component of the proxy user certificate to a text file:

```
$ keytool -export \
  -keystore config/proxy-user-keystore \
  -alias proxy-user-cert \
  -file config/proxy-user-cert.txt
```

5. Copy the `proxy-user-cert.txt` file into the `config` directory of all Directory Server instances. Import that certificate into each server's primary trust store by running the following command from the server root. When prompted for the keystore password, enter the password contained in the `config/truststore.pin` file. When prompted to trust the certificate, enter **yes**.

```
$ keytool -import \
  -keystore config/truststore \
  -alias proxy-user-cert \
  -file config/proxy-user-cert.txt
```

6. Update the configuration for each Directory Proxy Server instance to create a new key manager provider that will obtain its certificate from the `config/proxy-user-keystore` file. Run the following `dsconfig` command:

```
$ dsconfig create-key-manager-provider \
  --provider-name "Proxy User Certificate" \
  --type file-based \
  --set enabled:true \
  --set key-store-file:config/proxy-user-keystore \
  --set key-store-type:JKS \
  --set key-store-pin-file:config/proxy-user-keystore.pin
```

7. Update the configuration for each LDAP external server in each Directory Proxy Server instance to use the newly-created key manager provider, and also to use SASL EXTERNAL authentication instead of LDAP simple authentication. Run the following `dsconfig` command:

```
$ dsconfig set-external-server-prop \
  --server-name ds1.example.com:636 \
  --set authentication-method:external \
  --set "key-manager-provider:Proxy User Certificate"
```

After these changes, the Directory Proxy Server should re-establish connections to the LDAP external server and authenticate with SASL EXTERNAL. Verify that the Directory Proxy Server is still able to communicate with all backend servers by running the `bin/status` command. All of the servers listed in the "--- LDAP External Servers ---" section should be available. Review the Directory Server access log can to make sure that the BIND RESULT log messages used to authenticate the connections from the Directory Proxy Server include `authType="SASL"`, `saslMechanism="EXTERNAL"`, `resultCode=0`, and `authDN="cn=Proxy User,cn=Root DNs,cn=config"`.

Configuring Load Balancing

You can distribute the load on your Directory Proxy Server using one of the load-balancing algorithms provided with PingDirectoryProxy Server. By default, the Directory Proxy Server prefers local servers over non-local servers, unless you set the `use-location` property of the load-balancing algorithm to false. Within a given location, the Directory Proxy Server prefers available servers over degraded servers. This means that if at all possible, the Directory Proxy Server sends requests to servers that are local and available before considering selecting any server that is non-local or degraded.

Note: If the `use-location` property is set to true, then the load is balanced only among available external servers in the same location. If no external servers are available in the same location, the Directory Proxy Server will attempt to use available servers in the first preferred failover location, and so on. The failover based on no external servers with AVAILABLE health state can be customized to allow the Directory Proxy Server to prefer local DEGRADED health servers to servers in a failover location. See the PingDirectoryProxy Server Reference Guide for more information on the `prefer-degraded-servers-over-failover` property.

The Directory Proxy Server provides the following load-balancing algorithms:

- **Failover load balancing**. This algorithm forwards requests to servers in a given order, optionally taking the location into account. If the preferred server is not available, then it will fail over to the alternate server in a predefined order. This balancing method can be useful if certain operations, such as LDAP writes, need to be forwarded to a primary external server, with secondary external servers defined for failover if necessary.
- **Fewest operations load balancing**. This algorithm forwards requests to the backend server with the fewest operations currently in progress and tends to exhibit the best performance.
- **Health weighted load balancing**. This algorithm assigns weights to servers based on their health scores and, optionally, their locations. For example, servers with a higher health check score will receive a higher proportion of the requests than servers with lower health check scores.

- **Single server load balancing**. This algorithm forwards all operations to a single external server that you specify.
- **Weighted load balancing**. This algorithm uses statically defined weights for sets of servers to divide load among external servers. External servers are grouped into weighted sets, the values of which, when added to all of the weighted sets for the load balancing algorithm, represent a percentage of the load the external servers should receive.
- **Criteria based load balancing**. This algorithm allows you to balance your load across a server topology depending on the types of operations received or the client issuing the request.

  For example, ds1 and ds2 are assigned to a weighted set named Set-80 and assigned the weight 80. The external servers ds3 and ds4 are assigned to the weighted set Set-20 and assigned the weight 20. When both sets, Set-80 and Set-20, are assigned to the load balancing algorithm, 80 percent of the load will be forwarded to ds1 and ds2, while the remaining 20 percent will be forwarded to ds3 and ds4.

To Configure Load Balancing Using dsconfig

1. Use the `dsconfig` tool to create and configure a load-balancing algorithm.

```
$ bin/dsconfig
```

   Specify the hostname, connection method, port number, and bind DN as described in previous procedures.
2. In the Directory Proxy Server main menu, enter the number associated with load-balancing algorithms.
3. Select an existing load-balancing algorithm to use as a template or select `n` to create a new load-balancing algorithm from scratch.

```
>>>>Choose how to create the new Load Balancing Algorithm:

n) new Load Balancing Algorithm created from scratch
t) use an existing Load Balancing Algorithm as a template
b) back
q) quit

Enter a choice [n]: n
```

4. Select the type of load-balancing algorithm that you want to create. Depending on type of algorithm you select, you will be guided through a series of configuration properties, such as providing a name and selecting an LDAP external server.

```
>>>> Select the type of Load Balancing Algorithm that you want to
create:

    1)  Failover Load Balancing Algorithm
    2)  Fewest Operations Load Balancing Algorithm
    3)  Health Weighted Load Balancing Algorithm
    4)  Single Server Load Balancing Algorithm
    5)  Weighted Load Balancing Algorithm

    ?)  help
    c)  cancel
    q)  quit

Enter choice [c]: 3
```

5. Review the configuration properties for your new load-balancing algorithm. If you are satisfied, enter `f` to finish.

Configuring Criteria-Based Load-Balancing Algorithms

You can configure alternate load-balancing algorithms that determine how they function according to request or connection criteria. These algorithms allow you to balance your load across a server topology depending on the types of operations received or the client issuing the request. They are called criteria-based load-balancing algorithms and are configured using at least one connection criteria or request criteria. For example, you can configure criteria-based load-balancing algorithms to accomplish the following:

- Route write operations to a single server from a set of replicated servers, to prevent replication conflicts, while load balancing all other operations across the full set of servers.
- Route all operations from a specific client to a single server in a set of replicated servers, eliminating errors that arise from replication latency, while load balancing operations from other clients across the full set of servers. This configuration is useful for certain provisioning applications that need to write and then immediately read the same data.

When a request is received, the proxying request processor first iterates through all of the criteria-based load-balancing algorithms in the order in which they are listed, to determine whether the request matches the associated criteria. If there is a match, then the criteria-based load-balancing algorithm is selected. If there is not a match, then the default load-balancing algorithm is used.

Preferring Failover LBA for Write Operations

An administrator can configure the Directory Proxy Server to use Criteria-Based Load-Balancing Algorithms to strike a balance between providing a consistent view of directory server data for applications that require it and taking advantage of all servers in a topology for handling read-only operations, such as search and bind. The flexible configuration model supports a wide range of criteria for choosing which Load-Balancing Algorithm to use for each operation. In most Directory Proxy Server deployments, using a Failover Load-Balancing Algorithm for at least ADD, DELETE, and MODIFY-DN operations if not for all types of write operations is recommended.

Each Proxying Request Processor configured in the Directory Proxy Server uses a Load-Balancing Algorithm to choose which Directory Server to use for a particular operation. The Load-Balancing Algorithm takes several factors into account when choosing a server:

The availability of the directory servers.

The location of the directory servers. By default Load-Balancing Algorithms prefer directory servers in the same location as the Directory Proxy Server.

Whether the Directory Server is degraded for any reason, such as having a Local DB Index being rebuilt.

The result of configured Health Checks. For instance, a server with a small replication backlog can be preferred over one with a larger backlog.

Recent operation routing history.

How these factors are used depends on the specific Load-Balancing Algorithm. The two most commonly used Load-Balancing Algorithms are the Failover Load-Balancing Algorithm and the Fewest Operations Load-Balancing Algorithm. These two algorithms are similar when determining which Directory Servers are the possible candidates for a specific operation. The algorithms use the same criteria to determine server availability and health, and by default they will prefer Directory Servers in the same location as the Directory Proxy Server. However, they differ in the criteria they use to choose between available servers.

The Failover Load-Balancing Algorithm will send all operations to a single server until it is unavailable, and then it will send all operations to the next preferred server, and so on. This algorithm provides the most consistent view of the topology to clients because all clients (at least those in the same location as the Directory Proxy Server) will see the same, up-to-date view of the data, but it leaves unused capacity in the failover instances since most topologies include multiple Directory Server replicas within each data center.

On the other hand, the Fewest Operations Load-Balancing Algorithm does the best job of efficiently distributing traffic among multiple servers since it chooses to send each operation to the server that has the fewest number of outstanding operations--that is, the server from the Directory Proxy Server's point of view that is the least busy. (Note: the Fewest Operations Load-Balancing Algorithm routes traffic to the least loaded server, which in a lightly-loaded environment can result in an imbalance since the first server in the list of configured servers is more likely to receive a request.) This algorithm naturally routes to servers that are more responsive as well as limiting the impact of servers that have become unreachable. However, this implies that consecutive operations that depend on each other can be routed to different Directory Servers, which can cause issues for some types of clients:

If two entries are added in quick succession where the first entry is the parent of the second in the LDAP hierarchy, then the addition of the child entry could fail if that operation is routed to a different Directory Server instance than the first ADD operation, and this happens within the replication latency.

Some clients add or modify an entry and then immediately read the entry back from the server, expecting to see the updates reflected in the entry.

In these situations, it is desirable to configure the <keyword keyref="PROXY_SERVER_BASE_NAME"/> to route dependent requests to the same server.

The server affinity feature (see *Configuring Server Affinity*) achieves this in some environments but not in all because the affinity is tracked independently by each Directory Proxy Server instance, and some clients send requests to multiple proxies. It is common for a client to not connect to the Directory Proxy Servers directly but instead to connect through a network load balancer, which in turn opens connections to the Directory Proxy Servers. Each individual client connection will be established to a single Directory Proxy Server so that operations on that connection will be routed to the same Directory Proxy Server, and server affinity configured within the Directory Proxy Server will ensure those operations will be routed to the same Directory Server. However, many clients establish a pool of connections that are reused across operations, and within this pool, connections will be established through the load balancer to different Directory Proxy Servers. Dependent operations sent on different connections could then be routed to different Directory Proxy Servers, and then on to different Directory Servers.

A Failover Load-Balancing Algorithm addresses this issue by routing all requests to a single server, but that leaves unused search capacity on the other instances. A Criteria Based Load-Balancing Algorithm enables the proxy to route certain types of requests (or requests from certain clients) using a different Load-Balancing Algorithm than the default. For instance, all write operations (i.e., ADD, DELETE, MODIFY, and MODIFY-DN) could be routed using a Failover Load-Balancing Algorithm, while all other operations (bind, search, and compare) use a Fewest Operations Load-Balancing Algorithm. And in addition, if there are clients that are particularly sensitive to reading entries immediately after modifying them, additional Connection Criteria can be specified to all operations from those clients using the Failover Load-Balancing Algorithm. Note that, routing all write requests to a single server in a location instead of evenly across servers does not limit the overall throughput of the system since all servers ultimately have to process all write operations either from the client directly or via replication.

Another benefit of using the Failover Load-Balancing Algorithm for write operations is reducing replication conflicts. The Ping Identity Directory Server follows the traditional LDAP replication model of eventual consistency. This provides very high availability for handling write traffic even in the presence of network partitions, but it can lead to replication conflicts. Replication conflicts involving modify operations can be automatically resolved, leaving the servers in a consistent state where each attribute on each entry reflects the most recent update to that attribute. However, conflicts involving ADD, DELETE, and MODIFY-DN operations cannot always be resolved automatically and can require manual involvement from an administrator. By routing all write operations (or at least ADD, DELETE, and MODIFY-DN operations) to a single server, replication conflicts can be avoided.

There are a few points to consider when using a Failover Load-Balancing Algorithm:

- When using the Failover Load-Balancing Algorithm in a configuration with multiple locations, the Load-Balancing Algorithm will fail over between local instances before failing over to servers in a remote location. The list of servers in the `backend-server` configuration property of the Load-Balancing Algorithm should be ordered such that preferred local servers should appear before failover local servers, but the relative order of servers in different locations is unimportant as the `preferred-failover-location` of the Directory Proxy Server's configuration is used to decide which remote location to fail over to. It is also advisable that the order of local servers match the `gateway-priority` configuration settings of the "Replication Server" configuration object on the Directory Server instances. This can reduce the WAN replication delay because the Directory Proxy Server will then prefer to send writes to the Directory Server with the WAN Gateway role, avoiding an extra hop to the remote locations.
- For Directory Proxy Server configurations that include multiple Proxying Request Processors, including Entry-Balancing environments, each Proxying Request Processor should be updated to include its own Criteria-Based Load-Balancing Algorithm.

To Route Operations to a Single Server

The following example shows how to extend a Directory Proxy Server's configuration to use a Criteria Based Load Balancing Algorithm to route all write requests to a single server using a Failover Load Balancing Algorithm. The approach outlined here can easily be extended to support alternate criteria as well as more complex topologies using multiple locations or Entry Balancing.

This example uses a simple deployment of a Directory Proxy Server fronting three Directory Servers: ds1.example.com, ds2.example.com, and ds3.example.com.

Once these configurations changes are applied, the Directory Proxy Server will route all write operations to ds1.example.com as long as it is available and then to ds2.example.com if it is not, while routing other types of operations, such as searches and binds, to all three servers using the Fewest Operations Load Balancing Algorithm.

1. First, create a location.

   ```
   dsconfig create-location --location-name Austin
   ```
2. Update the failover location for your server.

   ```
   dsconfig set-location-prop --location-name Austin
   ```
3. Set the location as a global configuration property.

   ```
   dsconfig set-global-configuration-prop --set location:Austin
   ```
4. Set up the health checks for each external server.

   ```
   dsconfig create-ldap-health-check \
   --check-name ds1.example.com:389_dc_example_dc_com-search-health-check \
   --type search --set enabled:true --set base-dn:dc=example,dc=com

   dsconfig create-ldap-health-check \
   --check-name ds2.example.com:389_dc_example_dc_com-search-health-check \
   --type search --set enabled:true --set base-dn:dc=example,dc=com

   dsconfig create-ldap-health-check \
   --check-name ds3.example.com:389_dc_example_dc_com-search-health-check \
   --type search --set enabled:true --set base-dn:dc=example,dc=com
   ```
5. Create the external servers.

   ```
   dsconfig create-external-server --server-name ds1.example.com:389 \
   --type ping identity-ds \
   --set server-host-name:ds1.example.com --set server-port:389 \
   --set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
   --set password:AADoPkhx22qpiBQJ7T0X4wH7 \
   --set health-check:ds1.example.com:389_dc_example_dc_com-search-health-check


   dsconfig create-external-server --server-name ds2.example.com:389 \
   --type ping identity-ds \
   --set server-host-name:ds2.example.com --set server-port:389 \
   --set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
   --set password:AAAoVqVYsEavey80T0QfR60I \
   --set health-check:ds2.example.com:389_dc_example_dc_com-search-health-check


   dsconfig create-external-server --server-name ds3.example.com:389 \
   --type ping identity-ds \
   --set server-host-name:ds3.example.com --set server-port:389 \
   --set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
   --set password:AADOkveb0TtYR9xpkVrNgMtF \
   --set health-check:ds3.example.com:389_dc_example_dc_com-search-health-check
   ```
6. Create a Load Balancing Algorithm for `dc=example,dc=com`.

   ```
   dsconfig create-load-balancing-algorithm \
   --algorithm-name dc_example_dc_com-fewest-operations \
   --type fewest-operations --set  enabled:true \
   --set backend-server:ds1.example.com:389 \
   --set backend-server:ds2.example.com:389 \
   ```

```
--set backend-server:ds3.example.com:389
```

7. Create a Request Processor for `dc=example,dc=com`.

```
dsconfig create-request-processor \
--processor-name dc_example_dc_com-req-processor \
--type proxying \
--set load-balancing-algorithm:dc_example_dc_com-fewest-operations
```

8. Create a Subtree View for `dc=example,dc=com`.

```
dsconfig create-subtree-view \
--view-name dc_example_dc_com-view \
--set base-dn:dc=example,dc=com \
--set request-processor:dc_example_dc_com-req-processor
```

9. Update the client connection policy for dc=example,dc=com.

```
dsconfig set-client-connection-policy-prop \
--policy-name default \
--add subtree-view:dc_example_dc_com-view
```

10. Create a new Request Criteria object to match all write operations.

```
dsconfig create-request-criteria \
--criteria-name any-write \
--type simple --set "description:All Write Operations" \
--set operation-type:add --set operation-type:delete \
--set operation-type:modify --set operation-type:modify-dn
```

11. Create a new Failover Load Balancing Algorithm listing the servers that should be included. Note the order that the servers are listed here is the failover order between servers.

```
dsconfig create-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-failover \
--type failover --set enabled:true \
--set backend-server:ds1.example.com:389 \
--set backend-server:ds2.example.com:389 \
--set backend-server:ds3.example.com:389
```

12. Tie the Request Criteria and the Failover Load Balancing Algorithm together into a Criteria Based Load Balancing Algorithm.

```
dsconfig create-criteria-based-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-write-traffic-lba \
--set "description:Failover LBA For All Write Traffic" \
--set request-criteria:any-write \
--set load-balancing-algorithm:dc_example_dc_com-failover
```

13. Update the Proxying Request Processor to use the Criteria Based Load Balancing Algorithm.

```
dsconfig set-request-processor-prop \
--processor-name dc_example_dc_com-req-processor \
--set criteria-based-load-balancing-algorithm:dc_example_dc_com-write-
traffic-lba
```

To Route Operations from a Single Client to a Specific Set of Servers

To create a type of server affinity, where all operations from a single client are routed to a specific set of servers, follow a similar process as in the previous use case. Instead of request criteria, configure connection criteria. These connection criteria identify clients that could be adversely affected by replication latency. These clients will use the Failover Load Balancing Algorithm rather than the default Fewest Operations Load Balancing Algorithm.

For example, an administrative tool includes a "delete user" function. If the application immediately re-queries the directory for an updated list of users, the just-deleted entry must not be included. To configure a criteria-based load balancing algorithm to support this use case, perform the following:

- Create a failover load balancing algorithm that lists the same set of servers as the existing fewest operation load balancing algorithm.
- Create connection criteria that match the clients for which failover load balancing should be applied, rather than fewest operations load balancing.
- Create a criteria-based load balancing algorithm that references the two configuration objects created in the previous steps.
- Assign the new load balancing algorithm to the proxying request processor.

The following procedure provides examples of each of these steps.

1. Create the new failover load balancing algorithm using `dsconfig` as follows:

```
dsconfig create-load-balancing-algorithm \
  --algorithm-name client_one_routing_algorithm \
  --type failover --set enabled:true \
  --set backend-server:east1.example.com:389 \
  --set backend-server:east2.example.com:389
```

2. To route operations from a single client to a single server in a set of replicated servers, create connection criteria using `dsconfig` as follows:

```
dsconfig create-connection-criteria \
  --criteria-name "Client One"  --type simple \
  --set included-user-base-dn:cn=Client One,ou=Apps,dc=example,dc=com
```

3. Configure a criteria-based load balancing algorithm and assign it to the proxying request processor. Use the load balancing algorithm and connection criteria created in the previous steps:

```
dsconfig create-criteria-based-load-balancing-algorithm \
  --algorithm-name dc_example_dc_com-client-operations \
  --set load-balancing-algorithm:dc_example_dc_com-failover \
  --set "request-criteria:Client One Requests"
```

4. Assign the new criteria-based load balancing algorithm to the proxying request processor using `dsconfig` as follows:

```
dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add criteria-based-load-balancing-agorithm:dc_example_dc_com-client-
operations
```

Understanding Failover and Recovery

Once a previously degraded or unavailable server has recovered, it should be eligible to start receiving traffic within the time configured for the health-check-frequency property, 30 seconds by default. However, failover and recovery also depend on the load-balancing algorithm in use.

The load-balancing algorithm provides an ordered list of servers to check, with the number of servers in the list based on the maximum number of retry attempts. The server checks to see if affinity should be used and, if so, whether an affinity is set for that load-balancing algorithm. If there is an affinity to a particular server and that server is classified as available, then that server will always be the first in the list.

Next, the Directory Proxy Server creates a two-dimensional matrix of servers based on the health check state (with available preferred over degraded and unavailable not considered at all) and location (with backend servers in the same location as the Directory Proxy Server most preferred, then servers in the first failover location, then the second, and so on). Within each of these sets, and ideally at least one server in the local data center is classified as available, the load-balancing algorithm selects the servers in the order of most preferred to least preferred based on whatever logic the load-balancing algorithm uses. The load-balancing algorithm keeps selecting servers until enough of them have been selected to satisfy the maximum number of possible retries.

The load-balancing algorithm includes a configuration option that allows you to decide whether to prefer location over availability and vice-versa. For example, is a local degraded server more or less preferred than a remote available server? By default, the algorithm will prefer available servers over degraded ones, even if it has to go to another data center to access them. You can change the load-balancing algorithm to try to stay in the same data center if at least one server is not unavailable.

The Directory Proxy Server does both proactive and reactive health checking. With proactive health checking, the Directory Proxy Server will periodically (by default, every 30 seconds), run a full set of tests against each backend server. The result of these tests will be usd to determine the overall health check state (available, degraded, or unavailable) and score (and integer value from 10 to 0). With reactive health checking, the Directory Proxy Server may kick off a lesser set of health checks against a server if an operation forwarded to that server did not complete successfully.

Proactive health checking can be used to promote and demote the health of a server, but reactive health checking can only be used to demote the health of a server. As a result, if a server is determined to be unavailable, then it will remain that way until a subsequence proactive health check determines that it has recovered. If a server is determined to be degraded, it may not become available until the next proactive health check, but it could be downgraded to unavailable by a reactive check if other failures are encountered against that server.

Both proactive and reactive health check assignments take effect immediately and will be considered for all subsequent requests routed to the load-balancing algorithm. If a a server is considered degraded, then it will immediately be considered less desirable than available servers in the same data center, and possibly less desirable than available servers in more remote data centers. If a server is considered unavailable, then it will not be eligible to be selected until it is reclassified as available or degraded.

Configuring Proxy Transformations

The PingDirectoryProxy Server provides proxy transformations to alter the contents of client requests as they are sent from the client to the LDAP external server. Proxy transformations can also be used to alter the responses sent back from the server to the client, including altering or omitting search result entries. The Directory Proxy Server provides the following types of data transformations:

- **Attribute mapping**. This transformation rewrites client requests so that references to one attribute type may be replaced with an alternate attribute type. The Directory Proxy Server can perform extensive replacements, including attribute names used in DNs and attribute names encoded in the values of a number of different controls and extended operations. For example, a client requests a `userid` attribute, which is replaced with `uid` before being forwarded on to the backend server. This mapping applies in reverse for the response returned to the client.
- **Default value**. This transformation instructs the Directory Proxy Server to include a static attribute value in search results being sent back to the client, in ADD requests being forwarded to an external server, or both. For example, a value of "marketing" for `businessCategory` could be returned for all search results under the base DN `ou=marketing,dc=example,dc=com`.
- **DN mapping**. This transformation rewrites client requests so that references to entries below a specified DN will be mapped to appear below another DN. For example, references to entries below `o=example.com` could be rewritten so that they are below `dc=example,dc=com` instead. The mapping applies in reverse for the response returned to the client.
- **Groovy scripted**. This custom transformation is written in Groovy and does not need to be compiled, though they use the Server SDK. These scripts make it possible to alter requests and responses in ways not available using the transformations provided with the Directory Proxy Server.
- **Suppress attribute**. This proxy transformation allows you to exclude a specified attribute from search result entries. It also provides the ability to reject add, compare, modify, modify DN, or search requests if they attempt to reference the target attribute.
- **Suppress entry**. This proxy transformation allows you to exclude any entries that match a specified filter from a set of search results. Search requests are transformed so that the original filter will be ANDed with a NOT filter containing the exclude filter. For example, if the suppression filter is "`(objectClass=secretEntry)`", then a search request with a filter of "`(uid=john.doe)`" will be transformed so that it has a filter of "`(&(uid=john.doe)(!(objectClass=secretEntry)))`".
- **Simple to external bind**. This proxy transformation may be used to intercept a simple bind request and instead process the bind as a SASL EXTERNAL bind. If the SASL EXTERNAL bind fails, then the original simple bind request may or may not be processed, depending on how you configure the server.

- **Third-party scripted**. This custom transformation is created using the Server SDK, making it possible to alter requests and responses in ways not available using the transformations provided with the Directory Proxy Server.

To Configure Proxy Transformations Using dsconfig

1. Use the `dsconfig` tool to create and configure a proxy transformation.

   ```
   $ bin/dsconfig
   ```
2. Enter the connection parameters (for example, hostname, port, bind DN and bind DN password).
3. In the Directory Proxy Server main menu, enter the number associated with proxy transformations. On the Proxy Transformation menu, enter the number to create a new proxy transformation.
4. Select the type of proxy transformation you want to create. In this example, we create an attribute mapping transformation. Then, enter a name for the new transformation.

   ```
    >>>> Enter a name for the Attribute Mapping Proxy Transformation that you
   want to create: userid-to-uid
   ```
5. Indicate whether you want the transformation to be enabled by default.

   ```
   Select a value for the 'enabled' property:

     1) true
     2) false

     ?) help
     c) cancel
     q) quit

   Enter choice [c]: 1
   ```
6. Specify the name of the client attribute that you want to remap to a target attribute. Note that this attribute must not be equal to the target attribute.

   ```
   Enter a value for the 'source-attribute' property: userid
   ```
7. Specify the name of the target attribute to which the client attribute should be mapped.

   ```
   Enter a value for the 'target-attribute' property: uid
   ```
8. The properties of your new proxy transformation are displayed. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the proxy transformation.

   ```
   Enter choice [b]: f
   ```

   The transformation now needs to be assigned to a request processor. To create an initial request processor, see the next section.

Configuring Request Processors

A request processor is responsible for handling client requests by passing the request through a load-balancing algorithm or one or more subordinate request processors. The request processor is also the Directory Proxy Server component that performs proxy transformations. You can create one of the following types of request processors:

- **Proxying request processor**. This request processor is responsible for passing allowed operations through a load balancing algorithm. Proxy transformations can be applied to requests and responses that are processed. Multiple servers may be configured to provide high availability and load balancing, and various transformations may be applied to the requests and responses that are processed.
- **Entry-balancing request processor**. This request processor is used to distribute entries under a common parent entry among multiple backend sets. A backend set is a collection of replicated directory servers that contain identical portions of the data. This request processor uses multiple, subordinate proxying request processors

to process operations and maintains in-memory indexes to speed the processing of specific search and modify operations.

- **Failover request processor**. This request processor performs ordered failover between subordinate proxying processors, sometimes with different behavior for different types of operations.

To Configure Request Processors Using dsconfig

1. Use the `dsconfig` tool to create and configure a request processor.

```
$ bin/dsconfig
```

2. Specify the hostname, connection method, port number, and bind DN as described in previous procedures.
3. In the Directory Proxy Server main menu, enter the number associated with Request Processor configuration and select the option to create a new Request Processor.
4. Select an existing request processor to use as a template for creating a new one or enter `n` to create one from scratch. In this example, we create a new proxying request processor from scratch. You will be required to choose an existing load balancing algorithm or create a new one to complete the create of the request processor. Below is the configuration of the proxying request processor after selection of the load balancing algorithm.

```
Property                     Value(s)
---------------------------------------------------------------
1)  description              -
2)  enabled                  true
3)  allowed-operation        abandon, add, bind, compare, delete,
                             extended, modify, modify-dn, search
4)  load-balancing-algorithm dc_example_dc_com-fewest-operations
5)  transformation           -
6)  referral-behavior        pass-through
7)  supported-control        account-usable, assertion,
                             authorization-identity,
                             get-authorization-entry,
                             get-effective-rights, get-server-id,
                             ignore-no-user-modification,
                             intermediate-client, manage-dsa-it,
                             matched-values, no-op, password-policy,
                             permissive-modify, post-read, pre-read,
                             proxied-authorization-v1,
                             proxied-authorization-v2,
                             real-attributes-only, retain-identity,
                             subentries, subtree-delete,
                             virtual-attributes-only
8)  supported-control-oid    -

?)  help
f)  finish - create the new Proxying Request Processor
d)  display the equivalent dsconfig arguments to create this object
b)  back
q)  quit
```

5. Review the configuration properties of the new request processor. If you are satisfied, enter `f` to finish. For the request processor to be used, it must be associated with a subtree view.

To Pass LDAP Controls with the Proxying Request Processor

If your deployment does not use entry balancing and requires the use of LDAP controls not defined in the request processor's supported-control property, configure the Directory Proxy Server to forward these controls correctly. This is done by configuring the `supported-control-oid` property to define the request OID of the LDAP control. The Directory Proxy Server updates the root DSE `supportedControl` attribute with the values entered for the `supported-control-oid` property.

Configuring Server Affinity

The Directory Proxy Server supports the ability to forward a sequence of requests to the same external server if specific conditions are met. This feature, called server affinity, is applied by the load balancing algorithms. The following server affinity methods are available in the Directory Proxy Server:

- **Client Connection**. Requests from the same Directory Proxy Server client connection are consistently routed to the same external server.
- **Client IP**. Directory Proxy Server client requests coming from the same client IP address are routed to the same external server.
- **Bind DN**. Requests from all client connections authenticated as the same bind DN are routed to the same external server.

For each algorithm, you can specify the set of operations for which an affinity will be established, as well as the set of operations for which affinity will be used. Affinity assignments have a time-out value so that they are in effect for some period of time after the last operation that may cause the affinity to be set or updated.

To Configure Server Affinity

In this example, we create a bind DN server affinity provider for any client requesting write operations to have subsequent requests, whether read or write, forwarded to the same external server. The affinity period will last for 30 seconds after the last write request.

1. Use the `dsconfig` tool to configure server affinity. Specify the hostname, connection method, port number, and bind DN as described in previous procedures.

```
$ bin/dsconfig
```

2. In the Directory Proxy Server main menu, enter the number associated with server affinity provider configuration
3. On the Server Affinity menu, enter the number corresponding to creating a new server affinity provider.
4. Enter a name for your new server affinity provider.

```
>>>> Enter a name for the Bind DN Server Affinity Provider
that you want to create: Affinity for Writing Applications
```

5. Indicate whether you want the server affinity provider to be enabled for use by the Directory Proxy Server. In this example, enter `1` to enable to the server affinity provider.
6. Next, configure the properties of the server affinity provider. For example, you can customize the types of operations for which affinity may be set and the types of operations for which affinity may be used, as well as the length of time for which the affinity should persist. This example illustrates the properties of the bind DN server affinity provider.

```
>>>> >>>> Configure the properties of the Bind DN Server Affinity Provider

        Property                  Value(s)
        -----------------------------------------------------------
    1)  description               -
    2)  enable                    true
    3)  affinity-duration         30 s
    4)  set-affinity-operation    add, delete, modify, modify-dn
    5)  use-affinity-operation    add, bind, compare, delete, modify,
                                  modify-dn, search

    ?)  help
    f)  finish - create the new Bind DN Server Affinity Provider
    d)  display the equivalent dsconfig arguments to create this object
    b)  back
    q)  quit

Enter choice [b]:
```

7. Review the properties of the server affinity provider. If you are satisfied, enter `f` to finish. Once defined, the affinity provider can now be assigned to a load balancing algorithm.

Configuring Subtree Views

You provide clients access to a specific portion of the DIT creating a subtree view and assigning it to a client connection policy. You can configure subtree views from the command line or using the Administrative Console.

When you create a subtree view, you provide the following information to configure its properties:

- Subtree view name
- Base DN managed by the subtree view
- Request processor used by the subtree view to route requests. If one does not exist already, you will create a new one.

To Configure Subtree View

1. Use `dsconfig` to configure a subtree view.
2. In the Directory Proxy Server main menu, enter the number associated with subtree view configuration
3. In the Subtree View menu, enter the number corresponding to creating a new subtree view.
4. Enter a name for the subtree view.
5. Enter the base DN of the subtree managed by this subtree view.

```
Enter a value for the ‘base-dn’ property:dc=example,dc=com
```

6. Select a request processor for this subtree view to route requests or make the appropriate selection to create a new one.

```
Select a Request Processor for the ‘request-processor’ property:

1) dc_example_dc_com-req-processor
2) Create a new Request Processor

?) help
c) cancel
q) quit

Enter choice [c]: 1
```

7. Review the properties of the subtree view. If you are satisfied, enter `f` to finish.

```
>>>> Configure the properties of the Subtree View
 >>>> via creating 'example.com' Subtree View

        Property           Value(s)
        --------------------------------------------------
    1)  description        -
    2)  base-dn            "dc=example,dc=com"
    3)  request-processor  dc_example_dc_com-req-processor

    ?)  help
    f)  finish - create the new Subtree View
    d)  display the equivalent dsconfig arguments to create
        this object
    b)  back
    q)  quit
```

Once configured, you can assign one or more subtree views to any client connection policies.

Configuring Client Connection Policies

Client connection policies help distinguish what portions of the DIT the client can access. They also enforce restrictions on what clients can do in the server. A client connection policy specifies criteria for membership based on information about the client connection, including client address, protocol, communication security, and authentication state and identity. The client connection policy, however, does not control membership based on the type of request being made.

Every client connection is associated with exactly one client connection policy at any given time, which is assigned to the client when the connection is established. The choice of which client connection policy to use will be reevaluated when the client attempts a bind to change its authentication state or uses the StartTLS extended operation to convert an insecure connection to a secure one. Any changes you make to the client connection policy do not apply to existing connections. The changes only apply to new connections.

Client connections are always unauthenticated when they are first established. If you plan to configure a policy based on authentication, you must define at least one client connection policy with criteria that match unauthenticated connections.

Once a client has been assigned to a policy, you can determine what operations they can perform. For example, your policy might allow only SASL bind operations. Client connection policies are also associated with one or more subtree views, which determine the portions of the DIT a particular client can access. For example, you might configure a policy that prevents users connecting over the extranet from accessing configuration information. The client connection policy is evaluated in addition to access control, so even a root user connecting over the extranet would not have access to the configuration information.

Understanding the Client Connection Policy

Client connection policies are based on two things:

- **Connection criteria**. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used in other instances when the server needs to perform matching based on connection-level properties, such as filtered logging. A single connection can match multiple connection criteria definitions.
- **Evaluation order index**. If multiple client connection policies are defined in the server, then each of them must have a unique value for the **evaluation-order-index** property. The client connection policies are evaluated in order of ascending evaluation order index. If a client connection does not match the criteria for any defined client connection policy, then that connection will be terminated.

If the connection policy matches a connection, then the connection is assigned to that policy and no further evaluation occurs. If, after evaluating all of the defined client connection policies, no match is found, the connection is terminated.

When a Client Connection Policy is Assigned

A client connection policy can be associated with a client connection at the following times:

- When the connection is initially established. This association occurs exactly once for each client connection.
- After completing processing for a StartTLS operation. This association occurs at most once for a client connection, because StartTLS cannot be used more than once on a particular connection. You also may not stop using StartTLS while keeping the connection active.
- After completing processing for a bind operation. This association occurs zero or more times for a client connection, because the bind request can be processed many times on a given connection.

StartTLS and bind requests will be subject to whatever constraints are defined for the client connection policy that is associated with the client connection at the time that the request is received. Once they have completed, then subsequent operations will be subject to the constraints of the new client connection policy assigned to that client connection. This policy may or may not be the same client connection policy that was associated with the connection before the operation was processed. That is, any policy changes do not apply to existing connections and will be applicable when the client reconnects.

All other types of operations will be subject to whatever constraints are defined for the client connection policy used by the client connection at the time that the request is received. The client connection policy assigned to a connection never changes as a result of processing any operation other than a bind or StartTLS. So, the server will not re-evaluate the client connection policy for the connection in the course of processing an operation. For example, the client connection policy will never be re-evaluated for a search operation.

Restricting the Type of Search Filter Used by Clients

You can restrict the types of search filters that a given client may be allowed to use to prevent the use of potentially expensive filters, like range or substring searches. You can use the `allowed-filter-type` property to provide a list of filter types that may be included in the search requests from clients associated with the client connection policy. This setting will only be used if search is included in the set of allowed operation types. This restriction will only be applied to searches with a scope other than `baseObject`, such as searches with a scope of `singleLevel`, `wholeSubtree`, or `subordinateSubtree`.

The `minimum-substring-length` property can be used to specify the minimum number of non-wildcard characters in a substring filter. Any attempt to use a substring search with an element containing fewer than this number of bytes will be rejected. For example, the server can be configured to reject filters like `"(cn=a*)"` and `"(cn=ab*)"`, but to allow `"(cn=abcde*)"`. This property setting will only be used if search is included in the set of allowed operation types and at least one of `sub-initial`, `sub-any`, or `sub-final` is included in the set of allowed filter types.

There are two primary benefits to enforcing a minimum substring length:

- Allowing very short substrings can require the server to perform more expensive processing. The search requires a lot more server effort to assemble a candidate entry list for short substrings because the server has to examine a lot more index keys.
- Allowing very short substrings makes it easier for a client to put together a series of requests to retrieve all the data from the server (a process known as "trawling"). If a malicious user wants to obtain all the data from the server, then it is easier to issue 26 requests like `"(cn=a*)"`, `"(cn=b*)"`, `"(cn=c*)"`, ..., `"(cn=z*)"` than if the user is required to do something like `"(cn=aaaaa*)"`, `"(cn=aaaab*)"`, `"(cn=aaaac*)"`, ..., `"(cn=zzzzz*)"`.

Defining Request Criteria

The client connection policy provides several properties that allow you to define the kinds of requests that it can issue. The `required-operation-request-criteria` property causes the server to reject any requests that do not match the referenced request criteria. The `prohibited-operation-request-criteria` property causes the server to reject any request that matches the referenced request criteria.

Setting Resource Limits

A client connection policy can specify resource limits, helping to ensure that no single client monopolizes server resources. The resource limits are applied in addition to any global configuration resource limits. In other words, a client connection policy cannot grant additional resources beyond what is set in the global configuration. If a client connection exceeds either a globally-defined limit or a policy limit, then it is terminated.

Note: The Directory Proxy Server’s global configuration can enforce limits on the number of concurrent connections that can be established in the following ways:

- Limit the total number of concurrent connections to the server.
- Limit the total number of concurrent connections from the same IP address.
- Limit the total number of concurrent connections authenticated as the same bind DN.

Defining the Operation Rate

You can configure the maximum operation rate for individual client connections as well as collectively for all connections associated with a client connection policy. If the operation rate limit is exceeded, the Directory Proxy Server may either reject the operation or terminate the connection. You can define multiple rate limit values, making it possible to fine tune limits for both a long term average operation rate and short term operation bursts. For example, you can define a limit of one thousand operations per second and one million operations per day, which works out to an average of less than twelve operations per second, but with bursts of up to one thousand operations per second.

Rate limit strings should be specified as a maximum count followed by a slash and a duration. The count portion must contain an integer, and may be followed by a multiplier of k (to indicate that the integer should be interpreted as thousands), m (to indicate that the integer should be interpreted as millions), or g (to indicate that the integer should be interpreted as billions). The duration portion must contain a time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), and may be preceded by an integer to specify a quantity for that unit.

For example, the following are valid rate limit strings:

1/s (no more than one operation over a one-second interval)
10K/5h (no more than ten thousand operations over a five-hour interval)
5m/2d (no more than five million operations over a two-day interval)

You can provide time units in many different formats. For example, a unit of seconds can be signified using s, sec, sect, second, and seconds.

Client Connection Policy Deployment Example

In this example scenario, we assume the following:

Two external LDAP clients are allowed to bind to the Directory Proxy Server.
Client 1 should be allowed to open only 1 connection to the server.
Client 2 should be allowed to open up to 5 connections to the server.

Defining the Connection Policies

We need to set a per-client connection policy limit on the number of connections that may be associated with a particular client connection policy. We have to define at least two client connection policies, one for each of the two clients. Each policy must have different connection criteria for selecting the policy with which a given client connection should be associated.

Because the criteria is based on authentication, we must create a third client connection policy that applies to unauthenticated clients, because client connections are always unauthenticated as soon as they are established and before they have sent a bind request. Plus, clients are not required to send a bind request as their first operation.

Therefore, we define the following three client connection policies:

Client 1 Connection Policy, which only allows client 1, with an evaluation order index of 1.
Client 2 Connection Policy, which only allows client 2, with an evaluation order index of 2.
Unauthenticated Connection Policy, which allows unauthenticated clients, with an evaluation order index of 3.

We define simple connection criteria for the Client 1 Connection Policy and the Client 2 Connection Policy with the following properties:

The `user-auth-type` must not include none, so that it will only apply to authenticated client connections.
The `included-user-base-dn` should match the bind DN for the target user. This DN may be full DN for the target user, or it may be the base DN for a branch that contains a number of users that you want treated in the same way.

To create more generic criteria that match more than one user, you could list the DNs of each of the users explicitly in the `included-user-base-dn` property. If there is a group that contains all of the pertinent users, then you could instead use the `[all|any|not-all|not-any]-included-user-group-dn` property to apply to all members of that group. If the entries for all of the users match a particular filter, then you could use the `[all|any|not-all|not-any]-included-user-filter` property to match them.

How the Policy is Evaluated

Whenever a connection is established, the server associates the connection with exactly one client connection policy. The server does this by iterating over all of the defined client connection policies in ascending order of the evaluation order index. Policies with a lower evaluation order index value will be examined before those with a higher evaluation order index value. The first policy that the server finds whose criteria match the client connection will be associated with that connection. If no client connection policy is found with criteria matching the connection, then the connection will be terminated.

So, in our example, when a new connection is established, the server first checks the connection criteria associated with the Client 1 Connection Policy because it has the lowest evaluation order index value. If it finds that the criteria do not match the new connection, the server then checks the connection criteria associated with the Client 2 Connection Policy because it has the second lowest evaluation order index. If these criteria do not match, the server finally checks the connection criteria associated with the Unauthenticated Connection Policy, because it has the third lowest evaluation order index. It finds a match, so the client connection is associated with the Unauthenticated Connection Policy.

After the client performs a bind operation to authenticate to the server, then the client connection policies will be re-evaluated. If client 2 performs the bind, then the Client 1 Connection Policy will not match but the Client 2 Connection Policy will, so the connection will be re-associated with that client connection policy. Whenever a connection is associated with a client connection policy, the server will check to see if the maximum number of client connections have already been associated with that policy. If so, then the newly-associated connection will be terminated.

For example, Client 1 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. Client 1 then sends a bind request. The determination of whether the bind operation is allowed is made based on the constraints defined in the Unauthenticated Connection Policy, because it is the client connection policy already assigned to the client connection. Once the bind has completed, then the server will reevaluate the client connection policy against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to the Client 1 Connection Policy.

Next, Client 2 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. When Client 2 sends a bind request, the operation is allowed based on the constraints defined in the Unauthenticated Connection Policy. Once the bind is complete, the client connection is evaluated against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria do not match, so the client 2 connection is evaluated against the connection criteria associated with Client 2 Connection Policy, because it has the next lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to Client 2 Connection Policy.

Client 1 sends a search request. The Client 1 Connection Policy is used to determine whether the search operation should be allowed, because this is the client connection policy assigned to the client connection for client 1. The connection is not reevaluated, before or after processing the search operation.

To Configure a Client Connection Policy Using dsconfig

1. Use the `dsconfig` tool to create and configure a client connection policy.

   ```
   $ bin/dsconfig
   ```

2. Enter the connection parameters to the server (for example, hostname, connection method, port, bind DN and bind DN password).
3. In the Directory Proxy Server main menu, enter the number associated with client connection policy configuration. Then enter the number to create a new client connection policy.

   ```
   >>>> Client connection policy menu

   What would you like to do?

     1) List existing client connection policies
     2) Create a new client connection policy
     3) View and edit an existing client connection policy
     4) Delete an existing client connection policy
   ```

```
  b) back
  q) quit

Enter choice [b]: 2
```

4. Enter `n` to create a new client connection policy from scratch.

```
>>>> Select an existing Client Connection Policy to use as a
template for the new Client Connection Policy configuration or
‘n’ to create one from scratch:

  1) default

  n) new Client Connection Policy created from scratch
  c) cancel
  q) quit
```

5. Enter a name for the new client connection policy.

```
Enter the 'policy-id' for the Client Connection Policy that you

want to create: new_policy
```

6. Indicate whether you want the policy to be enabled by default.

```
Select a value for the 'enabled' property:

  1) true
  2) false

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

7. Provide a value for the `evaluation-order-index` property. Client connection policies with a lower index will be evaluated before those with a higher index.

```
Enter a value for the 'evaluation-order-index' property: 2
```

8. The properties of your new client connection policy are displayed. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the client connection policy.

   Any changes that you make to the client connection policy do not apply to existing connections. They will only apply to new connections.

Configuring Globally Unique Attributes

The PingDirectoryProxy Server supports a Globally Unique Attributes feature that ensures uniqueness for any value defined for a set of attributes within a subtree view. You can also configure when the server checks for attribute conflicts, either prior to any add, modify, or modify DN change request (pre-commit) or after the successful completion of a change request (post-commit).

About the Globally Unique Attribute Plug-in

The Directory Proxy Server supports a Globally Unique Attribute Plug-in that prevents any value within a defined set of attributes to appear more than once in any entry for one or more subtree views. Administrators can also configure whether conflict validation should be checked before an add, modify, or modify DN request to one or more backend servers or after the change has successfully completed.

For example, if the `pre-commit-validation` property is enabled, the Globally Unique Attribute Plugin performs one or more searches to determine whether any entries conflict with the change (i.e., add, modify, or modify DN). If a conflict is detected, then the change request will be rejected. If the `post-commit-validation` property is enabled, after the change has been processed, the server performs one or more searches to determine if a conflict was created in multiple servers at the same time. If a conflict is detected in this manner, then an administrative alert will be generated to notify administrators of the problem so that they can take any manual corrective action.

Note: The Globally Unique Attribute plug-in will attempt to detect and/or prevent unique attribute conflicts for changes processed through this Directory Proxy Server, but it cannot detect conflicts introduced by changes applied by clients communicating *directly* with backend servers.

We recommend that the Unique Attribute plug-in be enabled for all backend servers with the same configuration, so that conflicts can be detected within individual backend server instances. However, the Unique Attribute plug-in alone may not be sufficient for cases in which the content is split across multiple sets of servers (e.g., in an entry-balanced environment or in proxy configurations with different branches on different servers).

The LDAP SDK uniqueness request control can be used for enforcing uniqueness on a per-request basis. See the LDAP SDK documentation and the `com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl` class for using the control. See the ASN.1 specification to implement support for it in other APIs.

In general, note the following points about pre-commit validation versus post-commit validation:

- Pre-commit validation is the only mechanism that can try to prevent conflicts. It will increase the processing time for add, modify, and modify DN operations because the necessary searches to look for conflicts happen before the update request is forwarded to any backend servers.
- Post-commit validation will only let you know (via administrative alert) about conflicts that already exist in the data. It can't prevent conflicts, but can allow you to deal with them in a timely manner. It also operates during the post-response phase, so it won't affect the processing time for the associated write operation.
- In most cases, pre-commit validation should be sufficient to prevent conflicts, although we recommend that you periodically run the `identify-unique-attribute-conflicts` tool to find any conflicts that may have arisen. If you want to mitigate any risks due to conflicts being generated by concurrent operations in different servers, then using both `pre-commit-validation` and `post-commit-validation` properties provides the best combination of preventing most conflicts in advance, and detecting and alerting about conflicts that arise from concurrent writes.

For more detailed information about the plug-in, see the *Directory Proxy Server Reference (HTML)*

To Configure the Globally Unique Attribute Plug-in

The following example shows how to configure the Globally Unique Attribute plug-in. The example defines an attribute set consisting of the `telephoneNumber` and `mobile` attributes within the "test-view" subtree view. The `multiple-attribute-behavior` property determines the scope of how attributes may differ among entries and is the same property for the Directory Server plug-in. The property is set to `unique-across-all-attributes-including-in-same-entry`, which indicates that the `telephone` and `mobile` attributes must be unique throughout the subtree view, even within an entry. The `pre-commit-validation` property ensures that the Globally Unique Attribute Plugin performs one or more searches to determine whether any entries conflict with the change (i.e., add, modify, or modify DN). If a conflict is detected, then the change request will be rejected.

Note that all configured attributes should be indexed for equality in all backend servers.

- Run `dsconfig` to create the Globally Unique Attribute plug-in. The server will check that any add, modify, or modify DN request does not conflict with any attribute values in the entries. If there is a conflict, the change request will be rejected.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Globally-Unique telephone and mobile" \
  --type globally-unique-attribute \
  --set enabled:true \
  --set type:telephoneNumber \
  --set type:mobile \
  --set subtree-view:test-view \
  --set multiple-attribute-behavior:unique-across-all-attributes-including-
in-same-entry \
  --set pre-commit-validation:all-available-backend-servers
```

Configuring the Global Referential Integrity Plug-in

The PingDirectoryProxy Server supports a global referential integrity plug-in mechanism that maintains DN references from a specified set of attributes to entries that exist in the server (e.g., between the members values of a static group and the corresponding user entries). The plug-in intercepts delete and modify DN operations and updates any references to the target entry. For a delete operation, any references to the target entry are removed. For modify DN operations, any references to the target entry are updated to reflect the new DN of the entry.

The plug-in is similar to the Directory Server Referential Integrity Plug-in but does not have an asynchronous mode. When enabled on the Directory Proxy Server, the client response will be delayed until the referential integrity processing is complete. For Directory Proxy Server deployments not using entry balancing and using Directory Server external servers, it is best to instead use the Referential Integrity Plug-in on the Directory Server.

An equality index must be defined on all attributes referenced within the Global Referential Integrity Plug-in across all external servers.

Sample Global Referential Integrity Plug-in

- Use `dsconfig` to configure the Global Referential Integrity plug-in. The plug-in ensures that the `member`, `uniqueMember`, and `manager` attributes maintain their DN references in the defined subtree views. Note that any attributes for which referential integrity should be maintained should have values which are DNs and should be indexed for equality in all backend servers.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Global Referential Integrity" \
  --type global-referential-integrity \
  --set "enabled:true" \
  --set "attribute-type:member" \
  --set "attribute-type:uniqueMember" \
  --set "attribute-type:manager" \
  --set "subtree-view:employee-view" \
  --set "subtree-view:groups-view"
```

Configuring an Active Directory Server Back-end

Configuring an Active Directory server back-end requires a `dsconfig` script. The following settings are required for an Active Directory server:

- `verify-credentials-method:bind-on-existing-connections`, and `authorization-method:rebind`

  Active Directory does not support `proxy-as`. Existing connections must be reused.
- `set max-connection-age:5m`, and `health-check-pooled-connections:true`

  Active Directory drops idle connections after 15 minutes. The proxy needs to refresh the connection pool in a shorter interval.

The following example `dsconfig` script configures two Active Directory servers (AD-SRV1 and AD-SRV2).

```
dsconfig set-ldap-health-check-prop --check-name "Consume Admin Alerts" \
  --reset use-for-all-servers

dsconfig set-trust-manager-provider-prop \
  --provider-name "Blind Trust" \
  --set enabled:true

dsconfig create-external-server --server-name AD-SRV1 --type active-directory
 \
  --set server-host-name:example.server \
  --set server-port:636 \
  --set bind-dn:cn=ProxyUser,dc=dom-ad2,dc=local \
  --set password:password --set connection-security:ssl \
  --set key-manager-provider:Null --set trust-manager-provider:"Blind Trust" \
  --set authorization-method:rebind \
  --set verify-credentials-method:bind-on-existing-connections \
  --set max-connection-age:5m \
  --set health-check-pooled-connections:true

dsconfig create-external-server --server-name AD-SRV2 --type active-directory
 \
  --set server-host-name:example.server \
  --set server-port:636 \
  --set bind-dn:cn=ProxyUser,dc=dom-ad2,dc=local \
  --set password:password \
  --set connection-security:ssl \
  --set key-manager-provider:Null \
  --set trust-manager-provider:"Blind Trust" \
  --set authorization-method:rebind \
  --set verify-credentials-method:bind-on-existing-connections \
  --set max-connection-age:5m \
  --set health-check-pooled-connections:true

dsconfig create-load-balancing-algorithm --algorithm-name AD-LBA \
  --type fewest-operations \
  --set enabled:true \
  --set backend-server:AD-SRV1 \
  --set backend-server:AD-SRV2 \
  --set use-location:false

dsconfig create-request-processor --processor-name AD-Proxy --type proxying \
  --set load-balancing-algorithm:AD-LBA

dsconfig create-subtree-view --view-name AD-View \
  --set base-dn:dc=dom-ad2,dc=local \
  --set request-processor:AD-Proxy

dsconfig set-client-connection-policy-prop --policy-name default \
  --set subtree-view:AD-View
```

Chapter

4

Managing Access Control

Topics:

The PingDirectoryProxy Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and how it applies to the Directory Proxy Server.

Overview of Access Control

The access control model uses access control instructions (ACIs), which are stored in the `aci` operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and affects the entry or any subentries within that branch of the directory information tree (DIT).

Access control instructions specifies four items:

- **Resources**. *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the `cn` or `userPassword` password.
- **Name**. *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."
- **Clients**. *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: `groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."`
- **Rights**. *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant `read` or `write` permission to a `telephoneNumber` attribute.

Key Access Control Features

The PingDirectoryProxy Server provides important access control features that provide added security for the Directory Proxy Server's entries.

Improved Validation and Security

The Directory Proxy Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Proxy Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed `aci` value will be rejected.

As an additional level of security, the Directory Proxy Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the `lockdown-mode` privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to an access control instruction not being enforced. When the problem has been corrected, the administrator can use the `leave-lockdown-mode` tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the server (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the PingDirectoryProxy Server, global ACIs are defined within the server configuration, in the `global-aci` property of configuration object for the access control handler. They can be viewed and managed using configuration tools like `dsconfig` and the Administrative Console.

The global ACIs available by default in the PingDirectoryProxy Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: `namingContexts`, `subschemaSubentry`, `supportedAuthPasswordSchemes`, `supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedLDAPVersion`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`.

- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

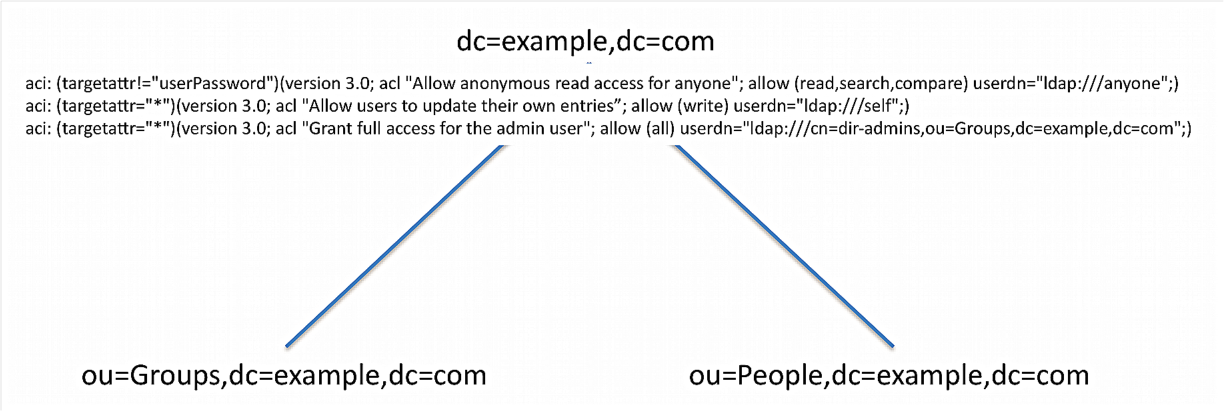
Access Controls for Public or Private Backends

The PingDirectoryProxy Server classifies backends as either public or private, depending on their intended purpose. A private backend is one whose content is generated by the Directory Proxy Server itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by the server (for example, the LDAP changelog backend). A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The PingDirectoryProxy Server access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General Format of the Access Control Rules

Access control instructions (ACIs) are represented as strings that are applied to one or more entries within the Directory Information Tree (DIT). Typically, an ACI is placed on a subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Directory Proxy Server iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissable operations. When a client application, such as `ldapsearch`, enters a request, the Directory Proxy Server checks that the user who binds with the server has the necessary access rights to the requested search targets. ACIs are cumulatively applied, so that a user who may have an ACI at an entry, may also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.



```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous read access for anyone"; allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")(version 3.0; acl "Allow users to update their own entries”; allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin user"; allow (all) userdn="ldap:///cn=dir-admins,ou=Groups,dc=example,dc=com";)
```

Figure 3: ACI

An access control rule has a basic syntax as follows:

```
aci : (targets) (version 3.0; acl "name"; permissions bind rules;)
```

Table 4: Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and/or attributes to which an access control rule applies. Syntax: *(target keyword = || != expression)*
name	Specifies the name of the ACI.
permissions	Specifies the type of operations to which an access control rule might apply. Syntax: *allow*||*deny (permission)*
bind rules	Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: *bind rule keyword =* ||*!= expression;*. The bind rule syntax requires that it be terminated with a ";".

Summary of Access Control Keywords

This section provides an overview of the keywords supported for use in the PingDirectoryProxy Server access control implementation.

Targets

A target expression specifies the set of entries and/or attributes to which an access control rule applies. The *keyword* specifies the type of target element. The *expression* specifies the items that is targeted by the access control rule. The operator is either the equal ("=") or not-equal ("!="). Note that the "!=" operator cannot be used with `targattrfilters` and `targetscope` keywords. For specific examples on each target keyword, see the section *Working with Targets*.

```
(keyword [=||!=]expression)
```

The following keywords are supported for use in the target portion of ACIs:

Table 5: Summary of Access Control Target Keywords

Target Keyword	Description	Wildcards
extop	Specifies the OIDs for any extended operations to which the access control rule should apply.	No
target	Specifies the set of entries, identified using LDAP URLs, to which the access control rule applies.	**Yes**
targattrfilters	Identifies specific attribute values based on filters that may be added to or removed from entries to which the access control rule applies.	**Yes**
targetattr	Specifies the set of attributes to which the access control rule should apply.	**Yes**
targetcontrol	Specifies the OIDs for any request controls to which the access control rule should apply.	No
targetfilter	Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.	**Yes**
targetscope	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI fi there is no target, to which the access control rule should apply.	No

Permissions

Permissions indicate the types of operations to which an access control rule might apply. You can specify if the user or group of users are allowed or not allowed to carry out a specific operation. For example, you would grant read access to a targeted entry or entries using "allow (read)" permission. Or you can specifically deny access to the target

entries and/or attributes using the "deny (read)" permission. You can list out multiple permissions as required in the ACI.

```
allow (permission1 ...,permission2,...permissionN)
```

```
deny (permission1 ...,permission2,...permissionN)
```

The following keywords are supported for use in the permissions portion of ACIs:

Table 6: Summary of Access Control Permissions

Permission	Description
add	Indicates that the access control should apply to add operations.
compare	Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
delete	Indicates that the access control should apply to delete operations.
export	Indicates that the access control should only apply to modify DN operations in which an entry is moved below a different parent by specifying a new superior DN in the modify DN request. The requestor must have the `export` permission for operations against the entry's original DN. The requestor must have the `import` permission for operations against the entry's new superior DN. For modify DN operations that merely alter the RDN of an entry but keeps it below the same parent (i.e., renames the entry), only the `write` permission is required. This is true regardless of whether the entry being renamed is a leaf entry or has subordinate entries.
import	See the description for the `export` permission.
proxy	Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identify to be specified).
read	Indicates that the access control rule should apply to search result entries returned by the server.
search	Indicates that the access control rule should apply to search operations with a non-base scope.
selfwrite	Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
write	Indicates that the access control rule should apply to modify and modify DN operations.
all	An aggregate permission that includes all other permissions except “proxy.” This is equivalent to providing a permission of “add, compare, delete, read, search, selfwrite, write, export, and import.”

Bind Rules

The Bind Rules indicate whether an access control rule should apply to a given requester. The syntax for the target keyword is shown below. The *keyword* specifies the type of target element. The expression specifies the items that is targeted by the access control rule. The operator is either equals ("=") or not-equals ("!="). The semi-colon delimiter symbol (";") is required after the end of the final bind rule.

```
keyword [=||!= ] expression;
```

Multiple bind rules can be combined using boolean operations (AND, OR, NOT) for more access control precision. The standard Boolean rules for evaluation apply: innermost to outer parentheses first, left to right expressions, NOT before AND or OR. For example, an ACI that includes the following bind rule targets all users who are not `uid=admin,dc=example,dc=com` and use simple authentication.

```
(userdn!="ldap:///uid=admin,dc=example,dc=com" and authmethod="simple");
```

The following bind rule targets the `uid=admin,dc=example,dc=com` and authenticates using SASL EXTERNAL or accesses the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or
 ip="127.0.0.1"));
```

The following keywords are supported for use in the bind rule portion of ACIs:

Table 7: Summary of Bind Rule Keywords

Bind Rule Keyword	Description

authmethod	Indicates that the requester's authentication method should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	`authmethod = method`
	where *method* is one of the following representations:
	none
	simple. Indicates that the client is authenticated to the server using a bind DN and password.
	ssl. Indicates that the client is authenticated with an SSL/TLS certificate (e.g., via SASL EXTERNAL), and not just over a secure connection to the server.
	sasl {sasl_mechanism}. Indicates that the client is authenticated to the server using a specified SASL Mechanism.
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (userdn="ldap:///self" and authmethod="ssl");)`

dayofweek	Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. Multiple day of week values may be separated by commas. The keyword's syntax is as follows:
	`dayofweek = day1, day2, ...`
	where *day* is one of the following representations:
	sun
	mon
	tues
	wed
	thu
	fri
	sat
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) on weekdays to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`

Bind Rule Keyword	Description

	`allow (write) (dayofweek!="sun,sat" and userdn="ldap:///`
	`self"`
	`and authmethod="ssl");)`

dns	Indicates that the requester's DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple DNS patterns may be separated by commas. The keyword's syntax is as follows:
	`dns` `=` *`dns-host-name`*
	The following example allows users on hostname `server.example.com` to update their own entries:
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write) (dns="server.example.com" and userdn="ldap:///`
	`self");)`

groupdn	Indicates that the requester's group membership should be taken into account when determining whether the access control rule should apply to any operation. Wildcards are not allowed in this expression.
	`groupdn [ = || != ] "ldap:///groupdn [ || ldap:///`
	`groupdn ] ..."`
	The following example allows users in the managers group to update their own entries:
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write)`
	`(groupdn="ldap:///`
	`cn=managers,ou=groups,dc=example,dc=com");)`

ip	Indicates that the requester's IP address should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple IP address patterns may be separated by commas. The keyword's syntax is as follows:
	`ip` `[ = || != ]` *`ipAddressList`*
	where *ipAddressList* is one of the following representations:
	A specific IPv4 address: 127.0.0.1
	An IPv4 address with wildcards to specify a subnetwork: 127.0.0.*
	An IPv4 address or subnetwork with subnetwork mask: 123.4.5.0+255.255.255.0
	An IPv4 address range using CIDR notation: 123.4.5.0/24
	An IPv6 address as defined by RFC 2373.
	The following example allows users on 10.130.10.2 and localhost to update their own entries:
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write) (ip="10.130.10.2,127.0.0.1" and`
	`userdn="ldap:///self");)`

Bind Rule Keyword	Description

timeofday	Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	`timeofday [ = || != || >= || > || <= || < ] time`
	where *time* is one of the following representations:
	4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour)
	Wildcards are not allowed in this expression
	The following example allows users to update their own entries if the request is received before 12 noon.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users who authenticate before noon`
	`    to update their own entries";`
	`    allow (write) (timeofday<1200 and userdn="ldap:///self"`
	`    and authmethod="simple");)`

userattr	Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of `USERDN` indicates that the target attribute should have a value which matches the DN of the authenticated user. A bindType value of `GROUPDN` indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A bindType value of `LDAPURL` indicates that the target attribute should have a value that is an LDAP URL whose criteria matches the entry for the authenticated user. Any value other than `USERDN`, `GROUPDN`, or `LDAPURL` is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:
	`userattr = attrName# [ bindType || attrValue ]`
	where:
	attrName = name of the attribute for matching
	bindType = USERDN, GROUPDN, LDAPURL
	attrValue = an attribute value. Note that the *attrVALUE* of the attribute must match on both the bind entry and the target of the ACI.
	The following example allows a manager to change employee's entries. If the bind DN is specified in the *manager* attribute of the targeted entry, the bind rule is evaluated to TRUE.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a manager to change employee`
	` entries";`
	`   allow (write) userattr="manager#USERDN";)`
	The following example allows any member of a group to change employee's entries. If the bind DN is a member of the group specified in the *allowEditors* attribute of the targeted entry, the bind rule is evaluated to TRUE.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow allowEditors to change employee`
	` entries";`
	`   allow (write) userattr="allowEditors#GROUPDN";)`

Bind Rule Keyword	Description

	The following example allows allows a user's manager to edit that user's entry and any entries below the user's entry up to two levels deep. You can specify up to five levels (0, 1, 2, 3, 4) below the targeted entry, with zero (0) indicating the targeted entry.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow managers to change employees entries`
	` two levels below";`
	`    allow (write) userattr="parent[0,1,2].manager#USERDN";)`
	The following example allows any member of the engineering department to update any other member of the engineering department at or below the specified ACI.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow any member of Eng Dept to update any`
	` other member of the`
	`   enginering department at or below the ACI";`
	`   allow (write) userattr="department#ENGINEERING";)`
	The following example allows an entry to be updated by any user whose entry matches the criteria defined in the LDAP URL contained in the `allowedEditorCriteria` attribute of the target entry.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a user that matches the filter to`
	` change entries";`
	`   allow (write) userattr="allowedEditorCriteria#LDAPURL";)`

userdn	Indicates that the user’s DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword’s syntax is as follows:
	`userdn [ = || != ] "ldap:///value [ || "ldap:///value ..."]`
	where *value* is one of the following representations:
	The DN of the target user
	A value of `anyone` to match any client, including unauthenticated clients.
	A value of `all` to match any authenticated client.
	A value of `parent` to match the client authenticated as the user defined in the immediate parent of the target entry.
	A value of `self` to match the client authenticated as the user defined in the target entry.
	If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.
	The following example allows users to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`    allow (write) userdn="ldap:///self";)`

Working with Targets

The following section presents a detailed look and examples of the target ACI keywords: `target`, `targetattr`, `targetfilter`, `targattrfilters`, `targetscope`, `targetcontrol`, and `extop`.

target

The `target` keyword indicates that the ACI should apply to one or more entries at or below the specified distinguished name (DN). The target DN must be equal or subordinate to the DN of the entry in which the ACI is placed. For example, if you place the ACI at the root of `ou=People,dc=example,dc=com`, you can target the DN, `uid=user.1,ou=People,dc=example,dc=com` within your ACI rule. The DN must meet the string representation specification of distinguished names, outlined in RFC 4514, and requires that special characters be properly escaped.

The `target` clause has the following format, where DN is the distinguished name of the entry or branch:

```
(target = ldap:///DN)
```

For example, to target a specific entry, you would use a clause such as the following:

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

Note that, in general, specifying a target DN is not recommended. It is better to have the ACI defined in that entry and omit the `target` element altogether. For example, although you can have `(target="ldap:///uid=john.doe,ou=People,dc=example,dc=com)` in any of the `dc=example,dc=com` or `ou=People` entries, it is better for it to be defined in the `uid=john.doe` entry and not explicitly include the `target` element.

The expression allows for the "not equal" (!=) operator to indicate that all entries within the scope of the given branch that do NOT match the expression be targeted for the ACI. Thus, the following expression targets all entries within the subtree that do not match `uid=john.doe`.

```
(target != ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

The `target` keyword also supports the use of asterisk (*) characters as wildcards to match elements within the distinguished name. The following target expression matches all entries that contains and begins with "john.d, " so that entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=People,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
```

The following target expression matches all entries whose DN begins with "john.d," and matches the `ou` attribute. Entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=asia-branch,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
```

Another example of a complete ACI targets the entries in the `ou=People,dc=example,dc=com` branch and the entries below it, and grants the users the privilege to modify all of their user attributes within their own entries.

```
aci:(target="ldap:///ou=People,dc=example,dc=com")
  (targetattr="*")
  (version 3.0; acl "Allow all the ou=People branch to modify their own
 entries";
 allow (write) userdn="ldap:///self";)
```

targetattr

The `targetattr` keyword targets the attributes for which the access control instruction should apply. There are four general forms that it can take in the PingDirectoryProxy Server:

- **(targetattr="*")**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **(targetattr="+")**. Indicates that the access control rule applies to all operational attributes. User attributes will not automatically be included in this set.
- **(targetattr="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies only to the named set of attributes.

- **(targetattr!="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as user attributes and operational attributes. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes, such as when the entry was created, last modified and by whom. Metadata can also include attributes specifying which password policy applies to the user, or overrided default constraints like size limit, time limit, or look-through limit for that user.

The PingDirectoryProxy Server distinguishes between these two types of attributes in its access control implementation. The Directory Proxy Server does not automatically grant any access at all to operational attributes. For example, the following clause applies only to user attributes and not to operational attributes:

```
(targetattr="*")
```

You can also target multiple attributes in the entry. The following clause targets the common name (cn), surname (sn) and state (st) attribute:

```
(targetattr="cn||sn||st")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

To include all user and all operational attributes, you use both symbols, as follows:

```
(targetattr="*||+")
```

If there is a need to target a specific operational attribute rather than all operational attributes, then it can be specifically included in the values of the `targetattr` clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or if you want to target all user attributes and a specific operational attribute, then you can use them in the `targetattr` clause, as follows:

```
(targetattr="*||ds-rlim-size-limit")
```

The following ACIs are placed on the `dc=example,dc=com` tree and allows any user anonymous read access to all entries except the `userPassword` attribute. The second ACI allows users to update their own contact information. The third ACI allows the `uid=admin` user full access privileges to all user attributes in the `dc=example,dc=com` subtree.

```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous
   read access for anyone"; allow (read,search,compare) userdn="ldap:///
anyone";)
aci: (targetattr="telephonenumber||street||homePhone||l||st")
  (version 3.0; acl "Allow users to update their own contact info";
   allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin
 user";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

An important note must be made when assigning access to user and operational attributes, which can be outlined in an example to show the implications of the Directory Proxy Server not distinguishing between these attributes. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of `uid` and `employeeNumber`. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a Directory Proxy Server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users including administrators granted full rights by access control rules, but excluding users with the `bypass-acl` privilege, allow them to edit any other attributes in their own entry including those excluded by rules like `uid` and `employeeNumber` in the example above, or add, modify, or delete any entries below his or her own entry.

Because the PingDirectoryProxy Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the `modify-acl` privilege.

Another danger in using the `(targetattr!="x")` pattern is that two ACIs within the same scope could have two different `targetattr` policies that cancel each other out. For example, if one ACI has `(targetattr!="cn||sn")` and a second ACI has `(targetattr!="userPassword")`, then the net effect is `(targetattr="*")`, because the first ACI inherently allows `userPassword`, and the second allows `cn` and `sn`.

targetfilter

The `targetfilter` keyword targets all attributes that match results returned from a filter. The `targetfilter` clause has the following syntax:

```
(targetfilter = ldap_filter)
```

For example, the following clause targets all entries that contain "ou=engineering" attribue:

```
(targetfilter = "(ou=engineering)")
```

You can only specify a single filter, but that filter can contain multiple elements combined with the OR operator. The following clause targets all entries that contain "ou=engineering," "ou=accounting," and "ou=marketing."

```
(targetfilter = "(|(ou=engineering)(ou=accounting)(ou=marketing)")
```

The following example allows the user, `uid=eng-mgr`, to modify the `departmentNumber`, `cn`, and `sn` attributes for all entries that match the filter `ou=engineering`.

```
aci:(targetfilter="(ou=engineering)")
  (targetattr="departmentNumber||cn||sn")
  (version 3.0; acl "example"; allow (write)
   userdn="ldap:///uid=eng-mgr,dc=example,dc=com";)
```

targattrfilters

The `targattrfilters` keyword targets specific attribute *values* that match a filtered search criteria. This keyword allows you to set up an ACI that grants or denies permissions on an attribute value if that value meets the filter criteria. The `targattrfilters` keyword applies to individual values of an attribute, not to the whole attribute. The keyword also allows the use of wildcards in the filters.

The keyword clause has the following formats:

```
(target = "add=attr1:Filter1 && attr2:Filter2... && attrn:FilterN,
del=attr1:Filter1 && attr2:Filter2 ... && attrN:FilterN" )
```

where

add represents the operation of adding an attribute value to the entry

del represents the operation of removing an attribute value from the entry

attr1, attr2... attrN represents the targeted attributes

filter1, filter2 ... filterN represents filters that identify matching attribute values

The following conditions determine when the attribute must satisfy the filter:

- When adding or deleting an entry containing an attribute targeted a `targattrfilters` element, each value of that attribute must satisfy the corresponding filter.
- When modifying an entry, if the operation adds one or more values for an attribute targeted by a `targattrfilters` element, each value must satisfy the corresponding filter. If the operation deletes one or more values for a targeted attribute, each value must satisfy the corresponding filter.
- When replacing the set of values for an attribute targeted by a `targattrfilters` element, each value removed must satisfy the delete filters, and each value added must satisfy the add filters.

The following example allows any user who is part of the `cn=directory server admins` group to add the `soft-delete-read` privilege.

```
aci:(targattrfilter="add=ds-privilege-name:(ds-privilege-name=soft-delete-
read)")
  (version 3.0; acl "Allow members of the directory server admins group to
 grant the
   soft-delete-read privilege"; allow (write)
   groupdn="ldap:///cn=directory server admins,ou=group,dc=example,dc=com";)
```

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is base to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
    (targetattr="supportedControl||supportedExtension||
     supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
    (version 3.0; acl "Allow users to view Root DSE Operational Attributes";
     allow (read,search,compare) userdn="ldap:///anyone")
```

targetcontrol

The `targetcontrol` keyword is used to indicate whether a given request control can be used by those users targeted in the ACI. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying control OIDs.

The following ACI example shows the controls required to allow an administrator to use and manage the Soft-Delete feature. The Soft Delete Request Control allows the user to soft-delete an entry, so that it could be undeleted at a later time. The Hard Delete Request Control allows the user to permanently remove an entry or soft-deleted entry. The Undelete Request Control allows the user to undelete a currently soft-deleted entry. The Soft-Deleted Entry Access Request Control allows the user to search for any soft-deleted entries in the server.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.22||
   1.3.6.1.4.1.30221.2.5.23||1.3.6.1.4.1.30221.2.5.24")
  (version 3.0; acl "Allow admins to use the Soft Delete Request Control,
   Hard Delete Request Control,Undelete Request Control, and
   Soft-deleted entry access request control";
   allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

extOp

The `extop` keyword can be used to indicate whether a given extended request operation can be used. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying extended request OIDs.

The following ACI allows the `uid=user-mgr` to use the Password Modify Request (i.e., OID=1.3.6.1.4.1.4203.1.11.1) and the StartTLS (i.e., OID=1.3.6.1.4.1.1466.20037) extended request OIDs.

```
aci:(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")
  (version 3.0; acl "Allows the mgr to use the Password Modify Request and
 StartTLS;
   allow(read) userdn="ldap:///uid=user-mgr,ou=people,dc=example,dc=com";)
```

Examples of Common Access Control Rules

This section provides a set of examples that demonstrate access controls that are commonly used in your environment. Note that to be able to alter access control definitions in the server, a user must have the `modify-acl` privilege as discussed later in this chapter.

Administrator Access

The following ACI can be used to grant any member of the "`cn=admins,ou=groups,dc=example,dc=com`" group to add, modify and delete entries, reset passwords and read operational attributes such as `isMemberOf` and password policy state:

```
aci: (targetattr="+")(version 3.0; acl "Administrators can read, search or
 compare operational attributes";
allow (read,search,compare) groupdn="ldap:///
cn=admins,ou=groups,dc=example,dc=com";)
aci: (targetattr="*")(version 3.0; acl "Administrators can add, modify and
 delete entries";
allow (all) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

Anonymous and Authenticated Access

The following ACI allow anonymous read, search and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. The authenticated user will inherit the privileges of the anonymous ACI. In addition, the authenticated user can change `userPassword`:

```
aci: (targetattr="objectclass || uid || cn || mail || sn || givenName")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Anyone can access names and email addresses of entries
 representing people";
allow (read,search,compare) userdn="ldap:///anyone";)
```

```
aci: (targetattr="departmentNumber || manager || isMemberOf")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Authenticated users can access these fields for entries
 representing people";
allow (read,search,compare) userdn="ldap:///all";)
aci: (targetattr="userPassword")(version 3.0; acl "Authenticated users can
 change password";
allow (write) userdn="ldap:///all";)
```

If no unauthenticated access should be allowed to the Directory Server, the preferred method for preventing unauthenticated, or anonymous access is to set the Global Configuration property `reject-unauthenticated-requests` to true.

Delegated Access to a Manager

The following ACI can be used to allow an employee's manager to edit the value of the employee's `telephoneNumber` attribute. This ACI uses the `userattr` keyword with a bind type of `USERDN`, which indicates that the target entry’s manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
(version 3.0; acl "A manager can update telephone numbers of her direct
 reports";
allow (read,search,compare,write) userattr="manager#USERDN";)
```

Proxy Authorization

The following ACIs can be used to allow the application "`cn=OnBehalf,ou=applications,dc=example,dc=com`" to use the proxied authorization v2 control to request that operations be performed using an alternate authorization identity. The application user is also required to have the `proxied-auth` privilege as discussed later in this chapter:

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com";)
```

Validating ACIs Before Migrating Data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Sun/Oracle encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the PingDirectoryProxy Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a PingDirectoryProxy Server.

To validate an access control instruction, the PingDirectoryProxy Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values, and all other attributes will be ignored.

To Validate ACIs from a File

The `validate-acis` tool can process data contained in an LDIF file. It will ignore all attributes except `aci`, and will ignore all entries that do not contain the `aci` attribute, so any existing LDIF file that contains access control rules may be used.

1. Run the `bin/validate-acis` tool (UNIX or Linux systems) or `bat\validate-acis` (Win dows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

   ```
   $ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-
   acis.ldif
   ```

   ```
   # Processing complete # Total entries examined: 1
   # Entries found with ACIs: 1
   # Total ACI values found: 3
   # Malformed ACI values found: 0
   # Other processing errors encountered: 0
   ```
2. Review the results by opening the output file. For example, the `validated-acis.ldif` file that was generated in the previous step reads as follows:

   ```
   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr!="userPassword")
     (version 3.0; acl "Allow anonymous read access for anyone";
       allow (read,search,compare) userdn="ldap:///anyone";)

   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr="*")
     (version 3.0; acl "Allow users to update their own entries";
       allow (write) userdn="ldap:///self";)

   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr="*")
     (version 3.0; acl "Grant full access for the admin user";
       allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
   ```
3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed ACI syntax is encountered.

   ```
   $ bin/validate-acis --ldifFIle test-acis.ldif --outputFile validated-
   acis.ldif \
     --onlyReportErrors
   ```

   ```
   # Processing complete
   # Total entries examined: 1
   # Entries found with ACIs: 1
   # Total ACI values found: 3
   # Malformed ACI values found: 0
   # Other processing errors encountered: 0
   ```

   The output file shows the following message:

   ```
   # The following access control rule is malformed or contains an unsupported
   # syntax: The provided string '(targetattr!=userPassword)(version 3.0; acl
   # "Allow anonymous read access for anyone"; allow (read,search,compare)
   # userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
   # Instruction (ACI) because it failed general ACI syntax evaluation
   dn: dc=example,dc=com
   aci: (targetattr!=userPassword)
     (version 3.0; acl "Allow anonymous read access for anyone";
       allow (read,search,compare) userdn="ldap:///anyone";)

   # The following access control rule is valid
   ```

```
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Validate ACIs in Another Directory Proxy Server

The `validate-acis` tool also provides the ability to examine ACIs in data that exists in another Directory Proxy Server that you are planning to migrate to the PingDirectoryProxy Server. The tool helps to determine whether the Ping Identity Server accepts those ACIs.

- To use it in this manner, provide arguments that specify the address and port of the target Directory Proxy Server, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

Migrating ACIs from Sun/Oracle to PingDirectory Server

This section describes the most important differences in access control evaluation between Sun/Oracle and the PingDirectory Server.

Support for Macro ACIs

Sun/Oracle provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the PingDirectory Server. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN Bind Rule

Sun/Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The PingDirectory Server does not support DSEE roles and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting Operational Attributes

The Sun/Oracle access control model does not differentiate between user attributes and operational attributes. With Sun/Oracle, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible for several significant security holes in which users are unintentionally given access to operational attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the PingDirectory Server, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except

`userPassword`, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like `targetattr="creatorsName||modifiersName"`. All operational attributes can be targeted using the `"+"` character. So, `targetattr="+"` targets all operational attributes but no user attributes and `targetattr="*||+"` targets all user and operational attributes.

Specification of Global ACIs

Both DSEE and PingDirectory Server support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the `aci` attribute. In the PingDirectory Server, global ACIs are managed with `dsconfig` referenced in the `global-aci` property of the Access Control Handler.

Defining ACIs for Non-User Content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the PingDirectory Server, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting Access to Controls and Extended Operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as "`oid={oid},cn=features,cn=config`" where `{oid}` is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control: "`oid=2.16.840.1.113730.3.4.3,cn=features,cn=config`".

In the PingDirectory Server, the "`targetcontrol`" keyword can be used to define ACIs that grant or deny access to controls. The "`extop`" keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for Malformed ACI Values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The PingDirectory Server is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the `lockdown-mode` privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.

Note: Consider running the `import-ldif` tool with the --rejectFile option so that you can review any rejected ACIs.

About the Privilege Subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you cannot restrict the capabilities assigned to that root user.

The PingDirectory Server offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server

configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying Unsupported ACIs

The PingDirectory Server provides a `validate-acis` tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the PingDirectory Server instance. When migrating data from a DSEE deployment into a PingDirectory Server instance, the `validate-acis` tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the PingDirectory Server.

For more information about using this tool, see *Validating ACIs Before Migrating Data*.

Working with Privileges

In addition to the access control implementation, the PingDirectoryProxy Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the PingDirectoryProxy Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available Privileges

The following privileges are defined in the PingDirectoryProxy Server.

Table 8: Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the `audit-data-security` tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the `bypass-acl` privilege but without the `config-read` privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is not evaluated for bind

Privilege	Description
	operations so that password policy evaluation will still occur when binding as a user with this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the `config-read` privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the Directory Proxy Server. The user is also required to have the `jmx-read` privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Proxy Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Proxy Server via the Java Management Extensions (JMX). The user is also required to have the `jmx-read` privilege. Note that currently all of the information exposed by the server over JMX is read-only.
ldif-export	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group.
ldif-import	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the `ldif-export` privilege.
lockdown-mode	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.
modify-acl	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.
password-reset	This privilege is required for one user to be allowed to change another user’s password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.
privilege-change	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the `ds-privilege-name` operational attribute.

Privilege	Description
proxied-auth	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
server-restart	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
server-shutdown	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
soft-delete-read	This privilege is required for a user to access a soft-deleted-entry.
stream-values	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
unindexed-search	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
update-schema	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges Automatically Granted to Root Users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the `ds-cfg-inherit-default-root-privileges` attribute with a value of FALSE in the entries for those root users.
- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the `default-root-privilege-name` property of the Root DN configuration object. By default, this set of privileges includes:

audit-data-security
backend-backup
backend-restore
bypass-acl
config-read
config-write
disconnect-client
ldif-export
lockdown-mode
manage-topology
metrics-read
modify-acl
password-reset

permit-get-password-policy-state-issues
privilege-change
server-restart
server-shutdown
soft-delete-read
stream-values
unindexed-search
update-schema

The privileges not granted to root users by default includes:

bypass-pw-policy
bypass-read-acl
jmx-read
jmx-write
jmx-notify
permit-externally-processed-authentication
permit-proxied-mschapv2-details
proxied-auth

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the `config-read`, `config-write`, and `privilege-change` privileges, as well as either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning Additional Privileges for Administrators

To allow access to the Tasks backend, set up a global ACI that allows access to members of an Administrators group as follows:

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*||+")
         (version 5.0; acl "Access to the tasks backend for administrators";
          allow (all) groupdn="ldap:///
          cn=admins,ou=groups,dc=example,dc=com";)'
```

Assigning Privileges to Normal Users and Individual Root Users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling Privileges

Although the privilege subsystem in the PingDirectoryProxy Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Chapter 5

Deploying a Standard Directory Proxy Server

Topics:

You can deploy PingDirectoryProxy Server in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates a standard deployment scenario.

Creating a Standard Multi-Location Deployment

In this example deployment, PingDirectoryProxy Server will be deployed in the data centers of two geographic locations: east and west. All LDAP external servers in this deployment are PingDirectory Servers. The directory servers in the eastern city are assigned to the location named east, and the directory servers in the western city are assigned to the location named west.

This example refers to four PingDirectory Server instances in two locations with replication of the `dc=example,dc=com` base DN enabled:

ds-east-01.example.com
ds-east-02.example.com
ds-west-01.example.com
ds-west-01.example.com

We will configure four Directory Proxy Server instances:

proxy-east-01.example.com
proxy-east-02.example.com
proxy-west-01.example.com
proxy-west-02.example.com

Overview of the Deployment Steps

In this deployment scenario, we will take the following steps:

- Install the first Directory Proxy Server in east location using the `setup` or `setup.bat` file included in the zip installation file.
- Use the `create-initial-proxy-config` tool to provide a proxy user bind DN and password, define locations for each of our data centers, and configure the LDAP external servers in these data centers.
- Test external server communications after initial setup is complete and test a simulated external server failure.
- Install the second proxy server in the east location using the `setup` or `setup.bat` file included in the zip installation file and copy the configuration of the first Directory Proxy Server using the configuration cloning feature.
- Install two Directory Proxy Server instances in the west location, which includes using the setup file and manually setting the location to west using the `dsconfig` command, as well as copying the configuration of the Directory Proxy Server using the configuration cloning feature.

After the proxy server has been configured and tested, we then provide a tour of the configuration of each of the proxy server components. These properties can be modified later as needed using the `dsconfig` tool.

Installing the First Directory Proxy Server

To begin with, we have the PingDirectoryProxy Server installation zip file. In this example, we plan to use SSL security, so we also have a keystore certificate database and a pin file that contains the private key password for the keystore. The keystore files are only necessary when using SSL or StartTLS.

In this deployment scenario, the keystore database is assumed to be a Java Keystore (JKS), which can be created by the keytool program. For more information about using the keytool, see the "Security Chapter" in the PingDirectory Server Administration Guide.

The PingDirectoryProxy directory contains the following:

```
root@proxy-east-01: ls
ExampleKeystore.jks   ExampleTruststore.jks ExampleKeystore.pin
PingDirectoryProxy-7.0.1.0-with-je.zip
```

The `ExampleKeystore.jks` keystore file contains the private key entry for the `proxy-east-01.example.com` server certificate with the alias `server-cert`. The server certificate, CA, and intermediate signing certificates are all contained in the `ExampleTruststore.jks` file. The password for `ExampleKeystore.jks` is defined in clear text in the corresponding pin file, though the name of the file need not match as it does in our example. The private key password in our example is the same as the password defined for the `ExampleKeystore.jks` keystore.

To Install the First Directory Proxy Server

1. Unzip the compressed archive file into the PingDirectoryProxy directory and move to this directory.

```
root@proxy-east-01: unzip -q PingDirectoryProxy-<version>-with-je.zip
root@proxy-east-01: cd PingDirectoryProxy
```

2. Because we are configuring SSL security, copy the keystore and pin files into the `config` directory.

```
root@proxy-east01: cp ../*Keystore* config/
root@proxy-east01: cp ../*Truststore* config/
```

3. Next, we install the first proxy server by running the `setup` tool on `proxy-east-01.example.com` as follows:

```
root@proxy-east01: ./setup --no-prompt --acceptLicense \
--ldapPort 389 --rootUserPassword pass \
--aggressiveJVMTuning --maxHeapSize 1g \
--enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickname server-cert \
--useJavaTrustStore config/ExampleTruststore.jks
```

New keystore password files are created in `config/keystore.pin`. The original file, `config/ExampleKeystore.pin`, is no longer needed.

4. If you are not using SSL or StartTLS, then the SSL arguments are not necessary as follows:

```
root@proxy-east01: ./setup --no-prompt --acceptLicense \
--ldapPort 389 --rootUserPassword pass \
--aggressiveJVMTuning --maxHeapSize 1g
```

Once you have installed the Directory Proxy Server, you can configure it using the `create-initial-proxy-config` tool as presented in the next section.

Configuring the First Directory Proxy Server

Once the Directory Proxy Server has been installed, it can be automatically configured using the `create-initial-proxy-config` tool. This tool can only be used once for this initial configuration, after which we will have to use `dsconfig` to make any changes to our proxy server configuration.

Configuring the Directory Proxy Server with the `create-initial-proxy-config` tool involves the following steps:

Providing a Directory Proxy Server base DN and password.
Defining locations for each of our data centers, east and west.
Configuring the LDAP external server in the east location.
Configuring the LDAP external servers in the west location.
Applying the changes to the Directory Proxy Server.

To Configure the First Directory Proxy Server

1. Once we have completed setup, we run the `create-initial-proxy-config` tool as follows:

```
root@proxy-east01: bin/create-initial-proxy-config
```

2. Provide the bind DN and password that the Directory Proxy Server will use to authenticate to the backend PingDirectory Server instances. The `create-initial-proxy-config` tool requires that the same bind

DN and password be used to authenticate to all of the backend servers. All Directory Proxy Server instances have identical proxy user accounts and passwords. If necessary, the proxy user account password can be defined differently for each external server using `dsconfig` after the `create-initial-proxy-config` tool has been executed.

3. Specify the type of external server communication security that will be used to communicate with the PingDirectory Server instances. For this example, enter the option for 'None'.
4. Specify the base DNs of the PingDirectory Server instances that the Directory Proxy Server will access. For this example, use `dc=example,dc=com`.
5. Enter any other base DNs of the PingDirectory Server instances that will be accessed through the proxy server. Because we are only using one proxy base DN, press **Enter** to finished.

Defining Locations

Next, we define our first location, east, to accommodate the servers in our deployment located on the East Coast of the United States.

To Define Proxy Locations

1. Continuing from the same `create-initial-proxy-config` session, enter a location name for the Directory Proxy Server. In this example, enter `east`, and then press **Enter**.
2. Define a location named `west` for the servers in our deployment located on the West Coast. Press **Enter** when finished.
3. Select the location that contains the Directory Proxy Server itself. The Directory Proxy Server is located in the east.

Configuring the External Servers in the East Location

Once the locations have been defined, we need to identify the directory servers. First, we define one of the servers in the east location.

To Configure the External Servers in the East Location

1. Define one of the servers in the east location by entering the host name and port of the server. For this example, enter `ds-east-01.example.com:389`.

```
>>>> External Servers

External Servers identify directory server instances including
host, port, and authentication information.

Enter the host and port (host:port) of the first directory server
in 'east'

    b)  back
    q)  quit

Enter a host:port or choose a menu item [localhost:389]: ds-
east-01.example.com:389
```

2. Enter the option to prepare the server and all subsequent servers. Preparing the servers involves testing the connections to these servers and sets up the `cn=Proxy User` account on the Directory Proxy Server.
3. Enter the DN of the account with which to manage the `cn=Proxy User,cn=Root DNs,cn=config` account. For this example, use the default, `cn=Directory Manager`.
4. Repeat the previous steps to prepare the other server in the east location, ds-east-02.example.com.
5. Press **Enter** to complete preparing the servers.

To Configure the External Servers in the West Location

The same process used for the east location is used to define the LDAP external servers for the west location.

1. Define the first external server, ds-west-01.example.com.
2. Define the second server in the west location, ds-west-02.example.com.
3. Press **Enter** when finished.

Apply the Configuration to the Directory Proxy Server

Next, we review the configuration summary. Once we have confirmed that the changes are correct, we press **Enter** to write the configuration.

To Apply the Changes to the Directory Proxy Server

1. During the configuration process, the `create-initial-proxy-config` tool writes the configuration settings to a `dsconfig` batch file, which will then be applied to the Directory Proxy Server. The batch file can be reused to configure other servers. On the final step, the `create-initial-proxy-config` tool presents a configuration summary. Review the configuration and then apply the changes to the Directory Proxy Server. Press **Enter** to write the configuration to the server.
2. On the final confirmation prompt, press **Enter** to apply the changes to the proxy server, and then enter the LDAP connection parameters to the server. Once the changes have been applied, the `create-initial-proxy-config` tool cannot be used to configure this proxy server again.

Configuring Additional Directory Proxy Server Instances

We install and configure the second Directory Proxy Server by running the `setup` tool on proxy-east-02.example.com.

To Configure Additional Directory Proxy Server Instances

1. Copy the keystore and pin files into the `config` directory for the proxy-east-02.example.com server.

```
root@proxy-east-02: cp ../*Keystore* config/
root@proxy-east-02: cp ../*Truststore* config/
```

2. Install the second Directory Proxy Server by running the `setup` tool on proxy-east-02.example.com as follows:

```
root@proxy-east-02: ./setup --no-prompt \
--listenAddress proxy-east-02.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-east-02.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location east
```

3. Configure the third Directory Proxy Server, proxy-west-01.example.com in the same way as shown in the previous step. First, copy the keystore and pin files into the `config` directory.

```
root@proxy-west-01: cp ../*Keystore* config/
root@proxy-west-01: cp ../*Truststore* config/
```

4. Run the `setup` tool on proxy-west-01.example.com as follows:

```
root@proxy-west-01: ./setup --no-prompt \
--listenAddress proxy-west-01.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-west-01.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location west
```

5. Finally, repeat steps 3 and 4 to install the last Directory Proxy Server by first copying the keystore and pin files to the `config` directory and then running the `setup` command.

   At this point, all proxies have the same Admin Data backend and have the `all-servers` group defined as their configuration-server-group in the Directory Proxy Server Global Configuration object. When making a change to a Directory Proxy Server using the `dsconfig` command-line tool or the Administrative Console, you will have the choice to apply the changes locally only or to all proxies in the `all-servers` group.

Testing External Server Communications After Initial Setup

After setting up the basic deployment scenario, the communication between the proxies and the LDAP external servers can be tested using a feature in the proxy server in combination with an LDAP search.

To Test the External Communications After Initial Setup

After initial setup, the Directory Proxy Server exposes a special search base DN for testing external server connectivity, called the `backend server pass-through` subtree view. While disabled by default, you can enable this feature using `dsconfig` in the Client Connection Policy menu. Set the value of the `backend-server-passthrough-subtree-views` property to `TRUE`.

1. Run `dsconfig` to set the `include-backend-server-passthrough-subtree-views` property to `TRUE`.

```
root@proxy-east-01: dsconfig set-client-connection-policy-prop \
--policy-name default \
--set include-backend-server-passthrough-subtree-views:true
```

   Once set to true, an LDAP search against the Directory Proxy Server with the base DN `dc=example,dc=com,ds-backend-server=ds-east-02.example.com:389` instructs the Directory Proxy Server to perform the search against the `ds-east-02.example.com:389` external server with the base DN set to `dc=example,dc=com`. The value of `ds-backend-server` should be the name of the configuration object representing the external server. Depending on your naming scheme, this name may not be a `host:port` combination.

2. Run `ldapsearch` to fetch the `dc=example,dc=com` entry from the `ds-east-01.example.com` server. Perform this search on each external server to determine if external server communication has been configured correctly on the Directory Proxy Server.

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--baseDN "dc=example,dc=com,ds-backend-server=ds-east-01.example.com:389" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. You can also use this special subtree view to track the operations performed on each external server to help determine load balancing requirements. This LDAP search can be run with the base DN values for the `ds-east-01` and `ds-east-02` servers to track the distribution of search and bind requests over time. These statistics are reset to zero when the server restarts. The following example searches an external server's monitor entry to display operation statistics:

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=directory manager" \
--bindPassword password \
--baseDN "cn=monitor,ds-backend-server=ds-east-02.example.com:389" \
--searchScope sub --useStartTLS "(cn=ldap*statistics)"

dn: cn=LDAP Connection Handler 192.168.1.203 port 389
```

```
Statistics,cn=monitor,ds-backend-server=ds-east-02.example.com:389

objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-ldap-statistics-monitor-entry
objectClass: extensibleObject
cn: LDAP Connection Handler 192.168.1.203 port 389
Statistics
connectionsEstablished: 3004
connectionsClosed: 2990
bytesRead: 658483
bytesWritten: 2061549
ldapMessagesRead: 17278
ldapMessagesWritten: 22611
operationsAbandoned: 0
operationsInitiated: 17278
operationsCompleted: 14241
abandonRequests: 22
addRequests: 1
addResponses: 1
bindRequests: 3006
bindResponses: 3006
compareRequests: 0
compareResponses: 0
deleteRequests: 0
deleteResponses: 0
extendedRequests: 2987
extendedResponses: 2987
modifyRequests: 1
modifyResponses: 1
modifyDNRequests: 0
modifyDNResponses: 0
searchRequests: 8271
searchResultEntries: 8370
searchResultReferences: 0
searchResultsDone: 8246
unbindRequests: 2990
```

Testing a Simulated External Server Failure

Once you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the Directory Proxy Server redirects LDAP requests appropriately. In this procedure, we stop the ds-east-01.example.com:389 server instance and test searches through proxy-east-01.example.com.

To Test a Simulated External Server Failure

1. First, perform several searches against the Directory Proxy Server. Verify activity in each of the servers in the east location, ds-east-01 and ds-east-02, by looking at the access logs. Because we used the default load balancing algorithm of fewest operations, it is likely that all of the searches will go to only one of the proxies. The following simple search can be repeated as needed:

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

2. Next, stop the Directory Server instance on ds-east-01.example.com using the `stop-server` command and immediately retry the above searches. There should be no errors or noticeable delay in processing the search.

```
root@ds-east-01: bin/stop-server

root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. Restart the Directory Proxy Server instance on ds-east-01.example.com. Check the access log to confirm that the Directory Proxy Server started to include the ds-east-01 server in load-balancing within 30 seconds. The default time is 30 seconds, though you can change this default if desired.

Expanding the Deployment

In the following example deployment, the PingDirectory Server is deployed in a third, centrally-located data center. The directory servers in the central city is assigned to a new location named central. The proxies will use StartTLS to communicate with the directory servers in the central region.

Note: Other than the ability to add to the Directory Proxy Server's truststore, the `prepare-external-server` tool does not alter the Directory Proxy Server configuration in any way.

The Directory Proxy Server itself, installed on proxy-east-01.example.com, remains in the East location. This example will reconfigure load balancing between the six directory servers in three locations:

ds-east-01.example.com
ds-east-02.example.com
ds-west-01.example.com
ds-west-02.example.com
ds-central-01.example.com
ds-central-02.example.com

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Prepare the new external servers using the `prepare-external-server` tool.
- Use the `dsconfig` tool to configure the new LDAP external servers in the central data center and reconfigure the load-balancing algorithm to take these servers into account.
- Test external server communications after the servers have been configured and test a simulated external server failure.

Preparing Two New External Servers Using the prepare-external-server Tool

First, we prepare the external directory servers, ds-central-01 and ds-central-02, by creating the proxy user account and the supporting access rules. In this example, we will connect to the ds-central-01 PingDirectory Server using StartTLS. Because we are using StartTLS, we need to capture the ds-central-01 server's certificate and put it in the trust store on our Directory Proxy Server instance.

The `prepare-external-server` tool is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy. In this example, we run the tool on the ds-east-01 instance of the Directory Proxy Server.

To Prepare Two New External Servers Using the prepare-external-server Tool

1. Run the `prepare-external-server` tool to prepare the two new servers. On the first attempted bind to the server, the tool will report a "failed to bind" message as it cannot bind to the `cn=Proxy User` entry due to its not being created yet. The tool sets up the `cn=Proxy User` entry so that the Directory Proxy Server can access it and tests the communication settings to the server.

```
root@proxy-east-01: ./prepare-external-server \
--hostname ds-central-01.example.com --port 389 \
--baseDN dc=example,dc=com \
--proxyBindPassword password  \
--useStartTLS \
--proxyTrustStorePath ../config/ExampleTruststore.jks
```

```
Failed to bind as ‘cn=Proxy User’

Would you like to create or modify root user ‘cn=Proxy User” so that it is
available for this Directory Proxy Server? (yes / no)[yes]:

Enter the DN of an account on ds-central-01:389 with which to create or
 manage the ‘cn=Proxy User’
account [cn=Directory Manager]:

Enter the password for ‘cn=Directory Manager’:

Created ‘cn=Proxy User,cn=Root DNs,cn=config’
Testing ‘cn=Proxy User’ privileges ....Done
```

2. Repeat the process on the other new server in the central location, ds-central-02.

Note: For entry-balancing deployments, the global base DN is required when using `prepare-external-server`.

Adding the New PingDirectory Servers to the Directory Proxy Server

After preparing the external PingDirectory Servers to communicate with the Directory Proxy Server, we can now add the two servers in the central location to the proxy server instance. Because we have run the `prepare-external-server` tool, the two servers have the `cn=Proxy User` entry configured.

To Add the New PingDirectory Servers to the Directory Proxy Server

- Run the `dsconfig` tool, which is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy.

```
root@proxy-east-01:./dsconfig

>>>> Specify LDAP connection parameters

Directory Proxy Server hostname or IP address [localhost]:

How do you want to connect to the Directory Proxy Server at
localhost?

    1)  LDAP
    2)  LDAP with SSL
    3)  LDAP with StartTLS

Enter choice [1]: 1

Directory Proxy Server at localhost port number [389]:
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

Adding New Locations

First, we add a new central location, to which our new PingDirectory Servers will be added.

To Add a New Location

The following steps show how to add the new servers to a new location using `dsconfig` interactive.

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.

```
$ bin/dsconfig
```

2. On the main menu, enter the number corresponding to **Location**.
3. On the Location menu, enter the number corresponding to creating a new location.
4. Enter the option to create a new location from scratch.

5. Configure the `preferred-failover-location` property of the new location so that this location fails over first to the east location and then to the west location, should all of the servers in the central location become unavailable.
6. Add the east and west locations as values of the property, specifying them in the order that they will be used for failover.
7. Confirm that these are the correct values and finish configuring the location.

Editing the Existing Locations

Next, we edit the existing east and west locations to include the new central location in their failover logic. The new failover logic will be based on geographic distance, so that the east location will first fail over to central and then the west location.

To Edit Existing Locations

The following example procedure uses `dsconfig` interactive mode to edit the east location.

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. On the Directory Proxy Server console configuration menu, enter the number corresponding to Location.
3. On the Location menu, enter the number corresponding to viewing and editing an existing location. Then, enter the number corresponding to the Location to be changed.
4. Remove the west location from the `preferred-failover-location` property. It will be added later.
5. Add a new value to the `preferred-failover-location` property.
6. Select the values of the new failover locations for the east.
7. Confirm the new configuration information and save the changes.
8. Repeat steps 2-7 to reconfigure the failover logic for the west location to include the new central location.
9. List the locations to confirm that the new location was added correctly.

Adding New Health Checks for the Central Servers

Next, we must add new health checks for the two new servers.

To Add New Health Checks for the Central Servers

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. Select the number corresponding to creating a new health check.
3. Enter the option to use an existing health check as a template.
4. Enter the number corresponding to the ds-east-01 health check to use it as a template for the new health check.
5. Name the new health check using the same naming strategy established for the other servers in the deployment. As this health check is for the ds-central-01 server, the name takes the following format:

```
>>>> Enter a name for the Search LDAP Health Check that you want to create:
ds-central-01.example.com:389_dc_example_dc_com-search-health-check
```

6. Review the configuration properties and then enter `f` to finish configuring the new health check and save changes.
7. Repeat steps 2-6 to create another new health check for the `ds-central-02` server.

Adding New External Servers

Add new external servers by selecting “External Server” from the main menu.

To Add New External Servers

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. On the External Server menu, enter the number corresponding to "Create a new External Server".

3. Base the configuration of the new external server on the existing configuration of the ds-east-01 server. Enter `t` to use an existing External Server as a template.
4. Enter the number to base the configuration of the new server on the configuration of the ds-east-01 server.
5. Enter a name for the new ds-central-01 server that complies with the naming strategy.

   ```
   >>>> Enter a name for the Ping Identity DS External Server that you
   want to create: ds-central-01.example.com:389
   ```
6. Enter the value of the `server-host-name` property.
7. Review and modify the configuration properties of the external server.
8. On the External Server menu, change the `server-host-name` property to reflect the name of the `ds-central-01` server.
9. On the External Server menu, change the `location` property to reflect the central location.
10. Change the `health-check` property to reflect the new health check created for the ds-central-01 server in the previous section.
11. On the 'health-check' Property menu, enter the number to remove one or more values.
12. Add the health-check created in the previous section.
13. Select the health check associated with the ds-central-01 server.
14. Press **Enter** to use the value associated with ds-central-01 health check.
15. Review the configuration of the new external server and enter `f` to create the server.
16. Repeat these steps to add the new `ds-central-02` external server.

Modifying the Load Balancing Algorithm

To modify the existing load-balancing algorithm to include the newly created servers, select “Load-Balancing Algorithm” from the main menu.

To Modify the Load-Balancing Algorithm

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. Choose the option for Load-Balancing Algorithm.
3. On the Load-Balancing Algorithm menu, enter the number corresponding to "View and edit an existing Load-Balancing Algorithm".
4. Add the ds-central-01 and ds-central-02 servers to the `backend-server` configuration property.
5. On the backend-server property menu, enter the number corresponding to adding one or more values.
6. Select the external servers to add. In this example, select ds-central-01.example.com and ds-central-02.example.com.
7. Review the changes made to the load-balancing algorithm’s configuration properties, and enter `f` to save changes.

   The change has been saved and applied to the Directory Proxy Server. The load-balancing algorithm is referenced in the load-balancing-algorithm property of the request processor used by this Directory Proxy Server.
8. To view this property, go to the main menu and select the Request Processor option.
9. On the Request Processor menu, enter the number corresponding to view and edit an existing request processor.
10. Select the request process used by the Directory Proxy Server, and review the configuration properties.

    This request processor is used by the subtree view serviced by the Directory Proxy Server, which is in turn referenced by the client connection policy.

    Note: The changes made in this procedure are already in effect. The Directory Proxy Server does not have to be restarted.

Testing External Server Communication

After adding and configuring the new external servers, test the communication between the Directory Proxy Server and the LDAP external servers using the `include-backend-server-passthrough-subtree-views` property of the Directory Proxy Server in combination with an LDAP search. For more information about this option, see *Testing External Server Communications* on page 190.

To Test External Server Communication

- Run the `ldapseasrch` command to test communications on the ds-central-01 serverTask.

```
root@proxy-east-01: bin/ldapsearch --port 389 --bindDN "cn=directory
 manager" \
--bindPassword password \
--baseDN "dc=example,dc=com,ds-backend-server=ds-central-01.example.com:389"
 \
--searchScope base "(objectclass=*)"
```

You can repeat this search on the ds-central-02 server, to confirm that the server returns the entry as expected.

Testing a Simulated External Server Failure

Once you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the Directory Proxy Server redirects LDAP requests appropriately. We stop the ds-east-01.example.com:389 and ds-east-02.example.com:389 server instances and test searches through proxy-east-01.example.com.

To Test a Simulated External Server Failure

1. We stop the ds-east-01.example.com:389 and ds-east-02.example.com:389 server instances and test searches through proxy-east-01.example.com.
2. Perform several searches against the Directory Proxy Server. Verify activity in each of the servers in the east location, ds-east-01 and ds-east-02, by looking at the access logs. The following simple search can be repeated as needed:

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. Next, stop the Directory Server instance on ds-east-01.example.com and ds-east-02.example.com using the stop-server command and immediately retry the above searches. There should be no errors or noticeable delay in processing the search.

```
root@proxy-east-01: bin/stop-server

root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" --bindPassword password \
--baseDN "dc=example,dc=com" --searchScope base --useStartTLS \
"(objectclass=*)"
```

4. Check the access log to confirm that requests made to these servers are routed to the central servers, as these servers are the first failover location in the failover list for the ds-east-01 and ds-east-02 servers.
5. Restart the Directory Server instance on ds-east-01.example.com and ds-east-02.example.com. Check their access logs to ensure that traffic is redirected back from the failover servers.

Merging Two Data Sets Using Proxy Transformations

In the following example, the Example.com company acquires Sample Corporation. During the merger, Example.com migrates data from Sample's `o=sample` rooted directory, converting Sample's sampleAccount auxiliary object class usage to Example.com's exampleAccount object class for entries rooted under `dc=example,dc=com`. Knowing that it can take considerable time for Sample's directory clients to become aware of the new DIT and schema, proxy data transformations are created to give the Sample clients as consistent a view of the data as possible during the migratory period. These transformations allow the clients to search and modify entries under `o=sample` using the Sample Corp. schema.

Overview of the Attribute and DN Mapping

To achieve the merger of the two data sets, we create proxy transformations that map the Sample source attributes to Example.com target attributes as described in Table 9-1, “Attribute Mapping”. The Example.com schema already defines an attribute to contain the RDN of user entries, called `uid`. However, Example.com chooses to create two new attributes within its `exampleAccount` object class to accommodate two attributes in the Sample schema for representing the region and the DN of linked accounts.

During the merger, Example.com decides to re-parent Sample’s customer entries, which are defined under two different subtrees, ou=east,o=sample and ou=west,o=sample, placing them under Example.com’s `ou=people,dc=example,dc=com` subtree. Associated proxy transformations are described in Table 9-2, "DN Mappings". In this process, Example.com collapses the Sample tree, moving entries from the east and west region under a single DN, `dc=example,dc=com`. The DN proxy transformations assume that all Sample users have been co-located under this single Example.com subtree.

Table 9: Attribute Mapping

Sample Attribute	Example.com Attribute	Description
sampleID	uid	RDN of user entries
sampleRegion	exSampleRegion	String value representing the region
sampleLinkedAccounts	exSampleLinkedAccounts	DN value

Legacy Sample LDAP applications searching for entries in either the Sample base DN `ou=east,o=sample` or `ou=west,o=sample` will be successfully serviced, though there will be one or more differences in the user entries seen by the Sample legacy applications. Since the Example.com Directory Server has no knowledge of the Sample user’s former `ou=east` or `ou=west` association, search results for client searching under `o=sample` will return a DN that may differ from the original search base. For instance, a search for `sampleID=abc123` under `ou=west,o=sample` may return the user entry for abc123 with the DN of `sampleID=abc123,ou=east,o=sample`. The following table illustrates the mapping DNs.

Table 10: DN Mapping

Sample DN	Example.com DN
ou=east,o=sample	dc=example,dc=com
ou=west,o=sample	dc=example,dc=com
o=sample	dc=example,dc=com

About Mapping Multiple Source DNs to the Same Target DN

Some complications exist when defining multiple DN mappings that are used for the same request processor and the same source or target DN (or that have source or target DNs that are hierarchically related). The client request may not include enough information to disambiguate and determine the proper rule to follow.

Several solutions exist to avoid problems of disambiguation. If the client does not need to be able to see all mappings at the same time, then a new client connection policy can be created to use connection criteria that select the set of mappings applied to the client based on information such as the IP address or bind DN. Each client connection policy would have separated subtree views with separate proxying request processors that reference the appropriate transformation for that client.

Alternatively, if it is unnecessary to search under the o=sample base DN, then separate subtree views can be created in the same client connection policy. For example, one subtree view would be created for `ou=east,o=sample` and one for `ou=west,o=sample`. Each subtree view is then associated with its own proxying request processor, one for ou=east requests and one for ou=west requests.

An Example of a Migrated Sample Customer Entry

The following example is an example of a Sample customer entry that has been migrated to the Example.com database. The user entry is defined in the Example.com Directory Server's database as follows. The attributes that have undergone a proxy transformation are marked in bold. Note that this view is how the entry appears to search requests under the `dc=example,dc=com` base DN.

```
dn: uid=scase,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
exAccountNumber: 234098
exSampleRegion: east
exSampleLinkedAccounts: uid=jcase,ou=people,dc=example,dc=com
userPassword: password
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
l: Arlington
mail: sterlingcase@maildomain.com st: VA
```

The following examples shows what the Directory Proxy Server returns to LDAP clients who have requested the entry when searching under the o=sample base DN. Note that the DN returned includes ou=east, even though this branch does not exist in the Example.com DIT. It also returns the attribute names as they are defined in the Sample schema.

```
dn:  sampleID=scase,ou=east,o=sample
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
exAccountNumber: 234098
exSampleRegion: east
exSampleLinkedAccounts: sampleID=jcase,ou=people,dc=example,dc=com
userPassword: password
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
l: Arlington
mail: sterlingcase@maildomain.com st: VA
```

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Install any necessary schema on the Directory Proxy Server.
- Create three attribute mapping proxy transformations and three DN mapping proxy transformations
- Create a new proxying request processor, using the existing `dc_example_dc_com` request processor as a template.
- Assign the six proxy transformations to the new proxying request processor.

- Create a new subtree view for o=sample that references the new proxying request processor.
- Add the new subtree view to the existing client connection policy.
- Test our configuration by performing some searches on the Sample DIT.

About the Schema

The Directory Proxy Server inherits user-defined schema from all external servers by comparing `cn=schema` on these servers at Directory Proxy Server startup and at five minute intervals. As a result, example.com schema does not need to be added manually to the Directory Proxy Server's `config/schema` directory. We assume that the schema for Sample entries has been defined on the external servers with the example.com DIT, requiring no direct schema management on the Directory Proxy Server. The following schema definitions are assumed to exist on the external Directory Server:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.1
  NAME 'exAccountNumber'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.3
  NAME 'sampleLinkedAccounts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.2
  NAME 'sampleRegion'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.1
  NAME 'sampleID'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.3
  NAME 'exSampleLinkedAccounts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.2
  NAME 'exSampleRegion'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
objectClasses: ( 1.3.6.1.4.1.32473.2.2.1
  NAME 'exampleAccount'
  SUP top
  AUXILIARY
  MAY ( exAccountNumber $
        exSampleRegion $
        exSampleLinkedAccounts $
        sampleID $
        sampleRegion $
        sampleLinkedAccounts ) )
```

The schema file defines some Example.com schema, such as `exAccountNumber` and `exSampleRegion`, and some Sample schema, such as `sampleRegion` and `sampleID`.

Creating Proxy Transformations

We create three attribute mapping proxy transformations and three DN mapping proxy transformations. We run the `dsconfig` tool, which is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy.

To Create Proxy Transformations

1. In the main server root directory, PingDirectoryProxy, run the `start-server` command.

```
$ bin/start-server
```

2. Run `dsconfig` in interactive mode and enter the LDAP connection parameters.
3. On the Configuration main menu, enter the number corresponding to **Proxy Transformation**.

Creating the Attribute Mapping Proxy Transformations

Next, we create the attribute mapping proxy transformations using `dsconfig` interactive. We assume for this example that we are continuing from the previous `dsconfig` session. In the following example, this transformation maps `ou=east,o=sample` in the Sample schema `dc=example,dc=com` in the Example.com schema.

To Creating the Attribute Mapping Proxy Transformations

1. On the Proxy Transformation menu, enter the number corresponding to "Create a New Proxy Transformation".
2. Create a mapping from the sampleRegion attribute to the `exSampleRegion` attribute, enter the number corresponding to "Attribute Mapping Proxy Transformation".

```
>>>> Select the type of Proxy Transformation that you want to create:

    1)  Attribute Mapping Proxy Transformation
    2)  Default Value Proxy Transformation
    3)  DN Mapping Proxy Transformation
    4)  Groovy Scripted Proxy Transformation
    5)  Simple To External Bind Proxy Transformation
    6)  Suppress Attribute Proxy Transformation
    7)  Suppress Entry Proxy Transformation
    8)  Third Party Proxy Transformation
```

3. Enter a descriptive name for the new proxy transformation that illustrates the attribute mapping that it performs.
4. Press **Enter** to enable the proxy transformation.
5. Provide the name of the source attribute in the Sample schema to map to the Example.com schema, which is sampleRegion.
6. Review the configuration properties, and enter `f` to create the new attribute mapping proxy transformation.
7. Repeat the previous steps to create another attribute mapping proxy transformation. This time, map between the Sample Corporation’s sampleID attribute and the Example.com uid attribute.
8. Repeat the previous steps again to create a last attribute mapping proxy transformation, mapping between the Sample sampleLinkedAccounts attribute and the Example.com exSampleLinkedAccounts attribute.

Creating the DN Mapping Proxy Transformations

Now we create the DN mapping proxy transformations.

To Create the DN Mapping Proxy Transformations

1. On the Proxy Transformation menu, enter the number corresponding to Create a new Proxy Transformation.
2. Enter the option to create a new Proxy Transformation from scratch.
3. Enter the option for "DN Mapping Proxy Transformation."
4. Enter a name for the DN Mapping Proxy Transformation. This transformation maps `ou=east,o=sample` in the Sample schema `dc=example,dc=com` in the Example.com schema.
5. Select `TRUE` to enable the transformation by default.
6. Specify the source DN as it appears in client requests.

```
>>>> Configuring the 'source-dn' property

     Specifies the source DN that may appear in client
     requests which should be remapped to the target DN.
     Note that the source DN must not be equal to the target DN.
     Syntax: DN

Enter a value for the 'source-dn' property:
ou=east,o=sample
```

7. Specify the target DN, where requests for the source DN should be routed.

```
>>>> Configuring the 'target-dn' property

    Specifies the DN to which the source DN should be mapped.
    Note that the target DN must not be equal to the source
    DN.

    Syntax: DN

Enter a value for the 'target-dn' property: dc=example,dc=com
```

8. Review the configuration properties, and then enter f to create the new DN mapping proxy transformation.
9. using the previous steps, create a new DN mapping proxy transformation that maps `ou=west,o=sample` in the Sample schema to `dc=example,dc=com` in the Example.com schema, and name it `sample_west-to-example`.
10. Finally, create a DN mapping proxy transformation for the base DN of the Sample database.

Creating a Request Processor to Manage the Proxy Transformations

Next, we need to create a new proxying request processor that includes our new attribute and DN mapping proxy transformations. We will use the existing `dc_example_dc_com` request processor as a template.

To Create a Request Processor to Manage Proxy Transformations

1. On the Configuration main menu, enter the number corresponding to Request Processor.
2. On the Request Processor menu, enter the number corresponding to "Create a new Request Processor."
3. Choose the option to use the current request processor as a template.
4. Provide a name for the new proxying request processor, such as `o_sample-req-processor`.
5. Review the properties. The load-balancing algorithm is the same as for the previous request processor, though the transformation property must be changed. Enter the number corresponding to the `Transformation` property.
6. Enter the number corresponding to the proxy transformations that we created in the previous sections.
7. Select the attribute mapping proxy transformations first. Next, select the DN mapping proxy transformations. The order of the selection is important because we have related DNs. Begin with the DNs that are lower in the tree first, and finish with the base DN transformation.

```
Select the Proxy Transformations you wish to add:

    1)  sample-to-example       5)  sampleLinkedAccounts-to-
                                    exSampleLinkedAccounts
    2)  sample_east-to-example  6)  sampleRegion-to-
                                    exSampleRegion
    3)  sample_west-to-example  7)  Create a new Proxy
                                    Transformation
    4)  sampleID-to-uid         8)  Add all Proxy Transformations

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 4,5,6,2,3,1
```

8. Confirm that the proxy transformations are listed in the correct order and press **Enter** to accept and use the values.
9. Review the request processor properties, and enter `f` to save changes.

Creating Subtree Views

At this stage, we need to configure subtree views for the Directory Proxy Server.

To Create Subtree Views

1. On the Configuration main menu, enter the number corresponding to Subtree View.
2. On the Subtree View menu, enter the number corresponding to "Create a new Subtree View."
3. Enter the option to create the new subtree view from an existing one.
4. Select the `dc_example_dc_com-view` subtree view.
5. Enter a descriptive name for the subtree view configuration.
6. Configure the base DN property of the Sample dataset.
7. Enter the request processor created in the previous section.
8. Review the configuration properties, and enter `f` to save changes.

```
>>>> Configure the properties of the Subtree View

        Property           Value(s)
        -------------------------------------------------
    1)  description        -
    2)  base-dn            "o=sample"
    3)  request-processor  o_sample-req-processor

    ?)  help
    f)  finish - create the new Subtree View
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q) quit
```

Editing the Client Connection Policy

Finally, we edit the client connection policy to add our new `o=sample` subtree view.

To Edit the Client Connection Policy

1. On the Configuration main menu, enter the number corresponding to Client Connection Policy.
2. On the Client Connection menu, enter the number corresponding to "Create a new Client Connection."
3. In the configuration properties, select the `subtree-view` property. Enter the number corresponding to "Add one or more values" to add the new subtree view created for the previous example.
4. Select the subtree view that was created in the previous section.

```
Select the Subtree Views you wish to add:

    1)  o_sample-view
    2)  Create a new Subtree View
```

5. Review the subtree views now referenced by the property and press **Enter** to use these values.
6. Review the configuration properties of the client connection policy and enter `f` to save changes.

Testing Proxy Transformations

After setting up the deployment scenario, the Directory Proxy Server will now respond to requests to the `dc=example,dc=com` and o=sample base DNs. We now test the service by imitating example client requests to search and modify users.

Testing Proxy Transformations

The following example fetches the user with sampleID=scase under the `ou=east,o=sample` base DN.

1. Run `ldapsearch` to view a Sample entry.

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "ou=east,o=sample" "(sampleID=scase)"
   ```

   ```
   dn: sampleID=scase,ou=People,ou=east,o=sample
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   objectClass: exampleAccount
   objectClass: top
   description: A customer account migrated from Sample merger
   sampleID: scase
   userPassword: {SSHA}A5O4RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZXA==
   exAccountNumber: 234098
   givenName: Sterling
   cn: Sterling Case
   sn: Case
   telephoneNumber: +1 804 094 3356
   street: 00468 Second Street
   mail: sterlingcase@maildomain.com
   l: Arlington
   st: VA
   sampleRegion: east
   sampleLinkedAccounts: sampleID=jcase,ou=People,ou=east,o=sample
   ```
2. Modify the sampleRegion value, changing it to west. To do this, we first create a `ldapmodify` input file, called `scase-mod.ldif`, with the following contents:

   ```
   dn: sampleID=scase,ou=People,ou=east,o=sample
   changetype: modify
   replace: sampleRegion
   sampleRegion: west
   ```
3. Use the file as an argument in the `ldapmodify` command as follows.

   ```
   root@proxy-east-01: bin/ldapmodify --bindDN "cn=Directory Manager" \
   --bindPassword password --filename scase-mod.ldif
   ```

   ```
   Processing MODIFY request for sampleID=scase,ou=People, ou=east,o=sample
   MODIFY operation successful for DN sampleID=scase,ou=People,
    ou=east,o=sample
   ```
4. Search for scase's `sampleRegion` value under o=sample, we should see west:

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "o=sample"  "(sampleID=scase)" \
   sampleRegion
   ```

   ```
   dn: sampleID=scase,ou=People,ou=east,o=sample
   sampleRegion: west
   ```
5. Search for scase by `uid` rather than `sampleID`, under the `dc=example,dc=com` base DN. We see the Example.com schema version of the entry:

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "dc=example,dc=com"  "(uid=scase)"
   ```

   ```
   dn: uid=scase,ou=People,dc=example,dc=com
   objectClass: person
   objectClass: exampleAccount
   objectClass: inetOrgPerson
   objectClass: organizationalPerson
   objectClass: top
   ```

```
description: A customer account migrated from Sample merger
uid: scase
userPassword: {SSHA}A5O4RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZXA==
exAccountNumber: 234098
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
mail: sterlingcase@maildomain.com
l: Arlington
st: VA
exSampleRegion: west
exSampleLinkedAccounts: uid=jcase,ou=People,dc=example,dc=com
```

Chapter 6

Deploying an Entry-Balancing Directory Proxy Server

Topics:

You can deploy PingDirectoryProxy Server in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates an entry-balancing deployment scenario.

Deploying an Entry-Balancing Proxy Configuration

Entry-balancing is a Directory Proxy Server configuration that allows the entries within a portion of the Directory Information Tree (DIT) to reside on multiple external servers. This configuration is typically useful when the DIT contains many millions of entries, which can be difficult to bring completely into memory for optimal performance. Entry-balancing allows entries under a balancing point base DN to be divided among any number of separate directory servers, making the Directory Proxy Server responsible for intelligently routing requests based on the division.

In this example scenario, the entries in the DIT outside of the balancing point are replicated across all external servers known to the Directory Proxy Server. Replication on the external directory servers must be properly configured before proceeding through this example. The directory servers are expected to contain two replication domains: the global domain, `dc=example,dc=com`, and the balancing point, `ou=people,dc=example,dc=com`.

In this deployment scenario, an `austin-proxy1` instance of the Directory Proxy Server communicates with four external directory servers. The Directory Proxy Server is configured to use entry balancing for the `ou=people,dc=example,dc=com` base DN, with two sets of user entries split beneath it. The first set of user entries is defined in the replicated pair of external servers, `austin-set1.example.com` and `newyork-set1.example.com`. The second set of entries is defined in `austin-set2.example.com` and `newyork-set2.example.com`. The entries in the `dc=example,dc=com` DIT outside of the balancing point base DN are replicated among the four external servers.

The following `dsreplication status` output from the PingDirectory Server external servers describes the replication configuration that exists before creating the Directory Proxy Server configuration.

```
--- Replication Status for dc=example,dc=com: Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
--------:---------:--------:---------------------------:-------------

austin-set1.example.com:389  : 10003   : 0   : N/A  : 722087263
austin-set2.example.com:389  : 10003   : 0   : N/A  : 722087263
newyork-set1.example.com:389 : 10003   : 0   : N/A  : 722087263
newyork-set2.example.com:389 : 10003   : 0   : N/A  : 722087263

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset1):
 Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
--------:---------:---------:---------------------------:-----------------

austin-set1.example.com:389  : 100001  : 0  : N/A               : 178892712
newyork-set1.example.com:389 : 100001  : 0  : N/A               : 178892712

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset2):
 Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
---------:---------:---------:---------------------------:-----------------

austin-set2.example.com:389  : 100001  : 0       : N/A        : 1057593890
newyork-set2.example.com:389 : 100001  : 0       : N/A        : 1057593890
```

Determining How to Balance Your Data

If a single Directory Server instance can hold all of your data, then we recommend storing your data on a single server and replicating for high availability, as this simplifies your deployment. If a single server cannot hold all of your data, then you can spread it across multiple servers in several ways:

- If the data is already broken up by hierarchy and all of the clients understand how to access it that way, the number of top-level branches is small and a single Directory Server instance can hold all of the information within one or more branches. Configure the Directory Proxy Server with multiple base DNs and use simple load-balancing rather than entry balancing to simplify your deployment.

- If simply breaking up the data using the existing hierarchy is not an option, for example if a large number of top-level branches must be configured, then consider using entry balancing. The contents of any single branch still must fit on a given server, because only entries that are immediate subordinates of the entry-balancing base DN may be spread across multiple servers. Any entries that are further subordinates have to be placed in the same directory server instance as their parent.
- If one or more branches are so large that any single Directory Server instance cannot hold all of the data, you need to use entry balancing within that branch to divide the entries among two or more sets of Directory Servers. You may also need to change the way that the data is arranged in the server so that it uses as flat a DIT as possible, which is easier to use in an entry-balancing deployment.

In an entry-balancing deployment, there can be data that is common to all external directory servers outside the balancing point. This data is referred to as the global domain. The Directory Proxy Server entry-balancing configuration will contain at least two subtree views and associated request processors, one for the global domain and one for the entry-balancing domain. In our examples, the global domain is `dc=example,dc=com` and the entry-balancing domain is `ou=people, dc=example,dc=com`. The entry-balancing base DN, `ou=people,dc=example,dc=com`, is also the balancing point.

Entry Balancing and ACIs

In an entry-balancing deployment, access control instructions (ACIs) are still configured in the backend Directory Server data. When defining access controls in an entry-balancing deployment, you need to ensure that the data used by the access control rule is available for evaluation on all datasets.

If you use groups for access control and a group contains users from different data sets, then that group must exist on each dataset. For a single ACI to be applicable to entries in all datasets, it must be specified above the entry-balancing point. For example, if an ACI allows access to modify users that are part of group 1, then two things must exist on both data sets:

- Group 1 must exist in the `ou=groups` branch of both datasets.
- The ACI referencing group 1 must exist in the `ou=people` branch or above. The `ou=people` branch entry itself is part of the common data.

The Directory Proxy Server ensures that any changes to entries within the scope of the entry-balancing request processor, but outside the balancing point, are applied to all backend server sets. Any ACI stored at the entry-balancing point will be kept in sync if changes are made through the Directory Proxy Server.

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Install the Directory Proxy Server on `austin-proxy1`.
- Use the `create-initial-proxy-config` tool to provide our initial setup for entry balancing. The initial setup includes defining multiple subtree views and global indexes in support of entry balancing.
- Change the placement algorithm of the `austin-proxy-01` server to use an entry-count placement algorithm. This algorithm is used to select the backend set to which to forward an add request. It looks at the number of entries in the backend sets and forwards the add request to the backend with either the fewest or the most entries, depending on the configuration. You can also configure the placement algorithm to make the decision based on the on-disk database size rather than the number of entries.

Installing the Directory Proxy Server

We start by configuring the Directory Proxy Server. The four external servers, `austin-set1.example.com`, `newyork-set1.example.com`, `austin-set2.example.com`, and `newyork-set2.example.com`, are running.

To Install the Directory Proxy Server

- Run the `setup` program in non-interactive mode.

```
root@austin-proxy1: ./setup --acceptLicense \
--listenAddress austin-proxy1.example.com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --entryBalancing \
--aggressiveJVMTuning --maxHeapSize 2g --no-prompt
```

Configuring the Entry-Balancing Directory Proxy Server

Once the Directory Proxy Server has been installed, it can be automatically configured using the `create-initial-proxy-config` tool. This tool can only be used once for this initial configuration, after which we will have to use `dsconfig` to make any changes to our Directory Proxy Server configuration.

To Configure the Entry-Balancing Directory Proxy Server

1. Run the `create-initial-proxy-config` tool.

```
root@austin-proxy1: ./bin/create-initial-proxy-config
```

2. Our topology meets the requirements, press **Enter** to continue:

```
Some assumptions are made about the topology to keep
this tool simple:

1) all servers will be accessible via a single user account
2) all servers support the same communication security type
3) all servers are PingDirectoryProxy Servers

If your topology does not have these characteristics you can
use this tool to define a basic configuration and then use the
'dsconfig' tool or the Administrative Console to fine tune the
 configuration.

Would you like to continue? (yes / no) [yes]:
```

3. Provide the external server access credentials. All of our proxies have identical proxy user accounts and passwords.

```
Enter the DN of the proxy user account [cn=Proxy User,cn=Root
 DNs,cn=config]:

Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

4. Specify the type of security that the Directory Proxy Server will use to communicate with Directory Servers.
5. Enter a base DN of the Directory Server instances that will be accessed by the Directory Proxy Server.
6. Define the balancing point as a separate base DN, which is entry balanced:

```
Enter another base DN of the directory server instances that
will be accessed through the Directory Proxy Server:

    1)Remove dc=example,dc=com

    b)back

    q)quit

Enter a DN or choose a menu item [Press ENTER when finished
entering base DNs]: ou=people,dc=example,dc=com

Are entries within 'ou=people,dc=example,dc=com' split across
```

```
multiple servers so that each server stores only a subset of
the entries (i.e. is this base DN 'entry balanced')? (yes / no)
[no]: yes
```

7. In this example, the data in `ou=people,dc=example,dc=com` will be split across two backend sets. Enter 2 to specify that the data will be balanced across two sets of servers.

```
Across how many sets of servers is the data balanced?

    c) cancel creating ou=people,dc=example,dc=com
    q) quit

Enter a number greater than one or choose a menu item: 2
```

8. The balancing point is the same as our base DN, ou=people,dc=example,dc=com., so we use it as the entry balancing base.

```
>>>> Entry Balancing Base

The entry balancing base DN specifies the entry below which the
data is balanced. Entries not below this entry must be duplicated
in all the server sets. If all the entries in the base DN are
distributed the entry balancing base DN is the same as the base DN.

    c) cancel creating ou=people,dc=example,dc=com
    b) back
    q) quit

Enter the entry balancing base DN or choose a menu item
[ou=people,dc=example,dc=com]: ou=people,dc=example,dc=com
```

9. To improve the performance for equality search filters referencing the `uid` attribute, create a `uid` global index. Enter `yes` to add a new attribute to the global index.
10. Specify the `uid` attribute.

```
Enter attributes that you would like to add to the global index:

    c)cancel creating ou=people,dc=example,dc=com
    b)back
    q)quit

Enter an attribute name or choose a menu item [Press ENTER when
finished entering index attributes]: uid
```

11. To optimize Directory Proxy Server performance from the moment it starts accepting connections, enter the number corresponding to "Yes, and all subsequent attributes."
12. Press **Enter** to finish specifying index attributes.
13. Press **Enter** to enable RDN index priming.

```
Would you like to enable RDN index priming for
'ou=people,dc=example,dc=com'? (yes / no) [yes]:
```

14. Press **Enter** to finish specifying base DNs.

```
Enter another base DN of the directory server instances that
will be accessed through the Directory Proxy Server:

    1) Remove dc=example,dc=com
    2) Remove ou=people,dc=example,dc=com (distributed)

    b)  back
    q)  quit

Enter a DN or choose a menu item [Press ENTER when finished
```

```
entering base DNs]:
```

15. The external servers are spread among two locations, New York and Austin. This Directory Proxy Server instance is located in the austin location.

```
A good rule of thumb when naming locations is to use the
name of your data centers or the cities containing them.

    b)  back
    q)  quit

Enter a location name or choose a menu item: austin

    1)  Remove austin

    b)  back
    q)  quit
```

16. Define the newyork location:

```
Enter another location name or choose a menu item [Press ENTER
when finished entering locations]: newyork

    1)  Remove austin
    2)  Remove newyork

    b)  back
    q)  quit

Enter another location name or choose a menu item [Press ENTER
when finished entering locations]:
```

17. Select the austin location for this Directory Proxy Server instance:

```
Choose the location for this Directory Proxy Server

    1) austin
    2) newyork

    b) back
    q) quit

Enter choice [1]:
```

18. Specify the LDAP external server instances associated with this location.

```
Enter the host and port (host:port) of the first directory server
in 'austin'

     b)  back
     q)  quit

Enter a host:port or choose a menu item [localhost:389]:
austin-set1.example.com:389
```

19. Specify that the `austin-set1` server can handle requests from the global domain and from set 1 restricted domain.

```
Assign server austin-set1.example.com:389 to handle requests for
one or more of the defined sets of data:

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,2
```

20. Enter the number corresponding to "Yes, and all subsequent servers" to prepare the server for access by the Directory Proxy Server.

```
Would you like to prepare austin-set1.example.com:389 for access
by the Directory Proxy Server?

       1)Yes
       2)No
       3)Yes, and all subsequent servers
       4)No, and all subsequent servers

Enter choice [3]:
```

21. Select the entry-balanced data set that the austin-set1 server replicates with other servers.

```
You may choose a single entry-balanced data set with which
austin-set1.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 1
    2) None, data will not be replicated

Enter choice: 1

Testing connection to austin-set1.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ....Denied
```

22. Modify the root user for use by the Directory Proxy Server, specifying the directory manager password for the initial creation of the proxy user.

```
Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Enter the DN of an account on austin-set1.example.com:389
with which to create or manage the 'cn=Proxy User,cn=Root DNs,
cn=config' account and configuration [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':
Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config'privileges...Done
Setting replication set name .....
```

23. Since the replication set name has already been configured, we do not need to use the name created automatically by the Directory Proxy Server.

```
This server is currently configured for replication set 'dataset1'.
Would you like to reconfigure this server for replication set
'set-1'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done
Testing 'cn=Proxy User' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

24. Define the other Austin and New York servers using the same procedure as in the previous example:

```
Enter another server in 'austin'

    1) Remove austin-set1.example.com:389
    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]: austin-set2.example.com:389
```

```
Assign server austin-set2.example.com:389 to handle requests
for one or more of the defined sets of data

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,3

You may choose a single entry-balanced data set with which
austin-set2.example.com:389 will replicate data with other
servers

    1) ou=people,dc=example,dc=com; Server Set 2
    2) None, data will not be replicated

Enter choice: 1

Testing connection to austin-set2.example.com:389 ....Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Would you like to use the previously entered manager credentials
to access all prepared servers? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset2'.

Would you like to reconfigure this server for replication set 'set-2'?
(yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'austin'

    1) Remove austin-set1.example.com:389
    2) Remove austin-set2.example.com:389

    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:

>>>> >>>> Location 'newyork' Details
 >>>> External Servers

External Servers identify directory server instances including
host, port, and authentication information.

Enter the host and port (host:port) of the first directory server
in 'newyork':

    b) back
    q) quit
```

```
Enter a host:port or choose a menu item [localhost:389]:
newyork-set1.example.com:389

Assign server newyork-set1.example.com:389 to handle requests
for one or more of the defined sets of data

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,2

You may choose a single entry-balanced data set with which
newyork-set1.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 1
    2) None, data will not be replicated

Enter choice: 1

Testing connection to newyork-set1.example.com:389 ....Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset1'.

Would you like to reconfigure this server for replication set
'set-1'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'newyork'

    1) Remove newyork-set1.example.com:389
    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]: newyork-set2.example.com:389

Assign server newyork-set2.example.com:389 to handle requests
for one or more of the defined sets of data:

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,3

You may choose a single entry-balanced data set with which
new-york-set2.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 2
    2) None, data will not be replicated
```

```
Enter choice: 1

Testing connection to newyork-set2.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access.... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this Directory
Proxy Server? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config' Testing
'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset2'.
Would you like to reconfigure this server for replication
set 'set-2'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'newyork'

    1)Remove newyork-set1.example.com:389
    2)Remove newyork-set2.example.com:389

    b)back
    q)quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:

>>>> >>>> Configuration Summary

  External Server Security: None
  Proxy User DN: cn=Proxy User,cn=Root DNs,cn=config
  Location austin
    Failover Order: newyork
    Servers: austin-set1.example.com:389,
             austin-set2.example.com:389
  Location newyork
    Failover Order: austin
    Servers: newyork-set1.example.com:389,
             newyork-set2.example.com:389
  Base DN: dc=example,dc=com
    Servers: austin-set1.example.com:389,
             austin-set2.example.com:389,
             newyork-set1.example.com:389,
             newyork-set2.example.com:389
  Base DN:vou=people,dc=example,dc=com
    Entry Balancing Base: ou=people,dc=example,dc=com
    Server Set 1: austin-set1.example.com:389,
                  newyork-set1.example.com:389
    Server Set 2: austin-set2.example.com:389,
                  newyork-set2.example.com:389
    Index Attributes: uid (primed,unique)
    Prime RDN Index: Yes

    NOTE: The Directory Proxy Server must be restarted after
    this tool has completed to have index priming take place

       b) back
       q) quit
       w) write configuration

    Enter choice [w]
    >>>> Write Configuration

    The configuration will be written to a 'dsconfig' batch
    file that can be used to configure other Directory Proxy Servers.

    Writing Directory Proxy Server configuration to /proxy/dps-
cfg.txt.....Done
```

25. Enter `yes` to apply our configuration changes to the Directory Proxy Server.

```
Apply these configuration changes to the local Directory Proxy
Server? (yes /no) [yes]:

How do you want to connect to the Directory Proxy Server on localhost?

    1) LDAP
    2) LDAP with SSL
    3) LDAP with StartTLS

Enter choice [1]:

Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
Creating Locations ..... Done
Updating Failover Locations ..... Done
Updating Global Configuration ..... Done
Creating Health Checks ..... Done
Creating External Servers ..... Done
Creating Load-Balancing Algorithm for dc=example,dc=com .... Done
Creating Request Processor for dc=example,dc=com ..... Done
Creating Subtree View for dc=example,dc=com ..... Done
Updating Client Connection Policy for dc=example,dc=com ..... Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server
 Set 1 ..... Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set
 1...Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server
 Set 2 .... Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set
 2...Done
Creating Entry Balancing Request Processor for
 ou=people,dc=example,dc=com ..... Done
Creating Placement Algorithm for ou=people,dc=example,dc=com .... Done
Creating Global Attribute Indexes for ou=people,dc=example,dc=com ..... Done
Creating Subtree View for ou=people,dc=example,dc=com ..... Done
Updating Client Connection Policy for ou=people,dc=example,dc=com ..... Done

See /logs/create-initial-proxy-config.log for a detailed log of this
 operation

To see basic server configuration status and configuration you can launch /
bin/status
```

Configuring the Placement Algorithm Using a Batch File

Now, we configure the placement algorithm using a batch file. We want to place new entries added through the proxy via LDAP ADD operations into the least used dataset. We do this using an entry-count placement algorithm. To change the placement algorithm from round-robin to entry-count, we first create and enable an entry-count placement algorithm configuration object and then disable the existing round-robin placement algorithm. Our batch file, `dsconfig.post-setup`, contains the `dsconfig` commands required to create the entry-count placement algorithm and disable the old round-robin algorithm.

To Configure the Placement Algorithm Using a Batch File

The batch file contains comments to explain each `dsconfig` command. Note that in this example, line wrapping is used for clarity. The `dsconfig` command requires that the full command be provided on a single line.

The batch file itself looks like the following:

```
root@austin-proxy1:more ../dsconfig.post-setup

# This dsconfig operation creates the entry-count placement
# algorithm with the default behavior of adding entries to the
# smallest backend dataset first.

dsconfig create-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name entry-count --type entry-counter --set enabled:true

# Note that once the entry-count placement algorithm is created
# and enabled, we can disable the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then disable the
# original round-robin algorithm created during the setup
# procedure.

dsconfig set-placement-algorithm-prop
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin --set enabled:false

# At this point, LDAP ADD operations will be forwarded to an external
# server representing the dataset with the least number of entries.
```

- Run the `dsconfig` command using the batch file. Once the batch file has executed, a new entry-count placement algorithm, called entry-count, has been created, and the old round-robin placement algorithm, `round-robin`, has been disabled.

  ```
  root@austin-proxy1: bin/dsconfig --no-prompt \
  --bindDN "cn=directory manager" --bindPassword password \
  --port 389  --batch-file ../dsconfig.post-setup
  ```

  ```
  Batch file '../dsconfig.post-setup' contains 2 commands

  Executing: create-placement-algorithm --no-prompt
  --bindDN "cn=directory manager" --bindPassword pass
  --port 1389
  --processor-name ou_people_dc_example_dc_com-eb-req-processor
  --algorithm-name entry-count --type entry-counter --set enabled:true

  Executing: delete-placement-algorithm --no-prompt
  --bindDN "cn=directory manager" --bindPassword pass
  --port 1389
  --processor-name ou_people_dc_example_dc_com-eb-req-processor
  --algorithm-name round-robin --set enabled:false
  ```

Rebalancing Your Entries

If your deployment distributes entries using an entry counter placement algorithm or 3rd party algorithm, you may need to redistribute your entries. For example, imagine that you have an environment that distributes entries across three backends using an entry counter placement algorithm. This algorithm distributes entries to the backend that has the most space. Imagine that the backends all reach their maximum capacity and you decide to add a new backend to the deployment. You need to move the entries from the full backends and distribute them evenly across all the backends, including the new backend.

You might also want to deliberately rebalance your entries to meet the needs of your organization. For example, you can direct entry balancing based on attributes on the entries themselves. You can write a custom algorithm that looks at the value of an attribute that is being modified on the entry. Based on the attribute, you can then put this entry somewhere specific. You might use this feature if you want to have certain entries closer geographically to the client application using them. The geographical information could be included in the entry. Rebalancing would be used to move these entries to the server in the correct geographical location.

You can redistribute entry-balanced entries in two ways:

- **Using dynamic rebalancing.** With dynamic rebalancing, as existing entries get modified, they get moved. You configure dynamic rebalancing in the entry counter placement algorithm.
- **Using the `move-subtree` tool.** This tool can be used to move either small subtrees through a transactional method or to move large subtrees, potentially taking them offline for a short period.

The remainder of this section describes each of these method of entry rebalancing in more detail.

About Dynamic Rebalancing

During dynamic rebalancing entries get moved as they are modified. You configure dynamic rebalancing on the entry counter placement algorithm or a third-party placement algorithm that supports rebalancing. This algorithm keeps a count of the number of entries or the size of the backend set. You configure dynamic rebalancing using the following parameters:

- **rebalancing-enabled**. Determines whether entry rebalancing is enabled. When rebalancing is enabled, the placement algorithm is consulted after modify and add operations, to determine whether the target entry should be moved to a different backend set.
- **rebalancing-scope**. Indicates which modified entries are candidates for rebalancing. A value of top-level indicates that only entries immediately below the entry-balancing base can be rebalanced. A value of any indicates that entries at any level below the entry-balancing base may be rebalanced.
- **rebalancing-minimum-percentage**. Specifies the minimum threshold for entries to be migrated from one backend set to a preferred backend set with a smaller size. Entries are not migrated unless the percentage difference between the value of the current backend set and the value of the preferred backend set exceeds this threshold. This parameter prevents unnecessarily migrating entries back and forth between backend sets of similar sizes.
- **rebalancing-subtree-size-limit**. Specifies the maximum size of a subtree that can be rebalanced.
- **poll-interval**. Specifies how long to wait between polling the size of the backends to determine how to rebalance; works in conjunction with the `rebalancing-minimum-percentage` property.
- **placement-criteria**. Determines which approach to use to select a destination backend for rebalancing. Possible values are: `entry-count`, `backend-size`, or `custom`.

The following figure illustrates an entry-balancing base DN and three subtrees, A, B, and C. If the rebalancing scope is set to `any`, any child entries under the base DN can be rebalanced. For example, if a change is made to entry A1, the entire subtree A might be rebalanced, depending upon how you have configured rebalancing. If the rebalancing scope is set to `top-level`, rebalancing is only triggered when entries at the top level, such as A, are modified. Changes made to subentries, such as A1 or A2, do not trigger rebalancing. Rebalancing is also triggered upon the addition of entries such as `A1,A2`, provided the scope is `any`.

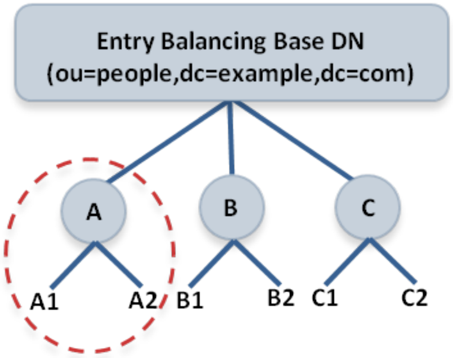


Figure 4: Rebalancing at the Top Level

If you are writing your own 3rd party algorithm, you program dynamic rebalancing using the SelectRebalancingBackendSet method on the placement algorithm. For more information, see the Server SDK documentation.

To Configure Dynamic Rebalancing

This procedure describes how to configure dynamic rebalancing on an existing entry balancing configuration.

1. To configure entry rebalancing, you may create an entry counter placement algorithm, if the current placement algorithm does not support rebalancing. You can either do this using `dsconfig` in interactive mode, or using the `dsconfig` command line as follows:

```
$ dsconfig create-placement-algorithm \
  --processor-name dc_example_dc_com-eb-req-processor \
  --algorithm-name rebalancing --type entry-counter \
  --set enabled:true --set rebalancing-enabled:true
```

2. Remove any placement algorithm previously configured on this entry-balancing request processor.
3. You can throttle how many entries are being moved by the proxy, so that the backend servers do not have too heavy a load. To do this, set the rebalancing-queue-maximum-size property of the request processor created in the previous step. By default, it is set to 1000. If the load is too high, reduce this value as follows:

```
$ dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set rebalancing-queue-maximum-size:50
```

4. Verify that the access logs are configured to display the subtrees being moved by dynamic rebalancing. The access logs provide a good way to monitor progress. So, if the write load on the backend servers is high and you see lots of rebalancing activity in the access log, lower the queue size to lower the rebalancing activity. You can configure the access log to display entry rebalancing processing information as follows:

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set log-entry-rebalancing-requests:true
```

About the move-subtree Tool

The `move-subtree` tool allows you to specify subtrees for rebalancing. You specify the source server, the target server, and one or more base DNs identifying the subtrees you want to move. You can move small subtrees using the transactional method or move large subtrees, which does not use this method. Instead, the large subtree is not fully accessible during the move, so clients may get an "insufficient access rights error" if they try to access the subtree. As entries are moved, clients can read but not write to them. Once the transfer is complete, the entries are fully available to client requests.

This tool accepts a file containing a list of the base DNs of the subtrees you want to move.

Note: The `move-subtree` tool requires users to have access to the extended operations and controls needed to run the tool. Make sure to apply the following ACIs to your data.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.5 ||
      1.3.6.1.4.1.30221.2.5.24 || 1.3.6.1.4.1.30221.2.5.13")
     (version 3.0; acl "Allow admin to submit move-subtree controls";
      allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (extop="1.3.6.1.4.1.30221.2.6.19")
     (version 3.0; acl "Allow admin to request move-subtree extended
      operation"; allow (read) userdn="ldap:///
uid=admin,dc=example,dc=com";)
```

About the subtree-accessibility Tool

The `subtree-accessibility` tool helps you determine if a subtree has restricted access and helps you fix any problems. If, during rebalancing, the Directory Server issues an alert that a subtree has been unavailable for too long, then you can use this tool to evaluate the problem. For example, if the `move-subtree` tool is interrupted by a host machine going down unexpectedly, the subtree might not be successfully moved. You can use the `subtree-accessibility` tool to evaluate and correct any problems with the subtrees, and then re-run the `move-subtree` tool.

Managing the Global Indexes in Entry-Balancing Configurations

In an entry-balancing configuration, the Directory Proxy Server maintains the default RDN index as well as one or more in-memory global attribute indexes. The global indexes allow the Directory Proxy Server to select the correct backend server set for incoming operations, which avoids broadcasting operations to all backend sets.

The indexes may be preloaded from peer proxies or the backend directory servers when the server starts up, and are updated by certain operations that come through the Directory Proxy Server. For instance, when a new entry is added, the DN of the new entry is added to the DN index of the Directory Proxy Server performing the operation. The indexes are also fault-tolerant and can adapt to changes made in the backend servers without going through the Directory Proxy Server. For example, operations will be processed directly through the backend server.

This section describes when to create a global attribute index, how to reload the global index, how to monitor its growth, and how to prime the global index from a peer at start-up.

When to Create a Global Attribute Index

The RDN index is referenced whenever a modify, delete, or base search is requested. In other words, the RDN index is needed when the LDAP request contains the complete DN of the targeted entry. If the entry-balancing request processor is not configured to prime the rdn index at startup, then the index is populated over time as LDAP requests are processed.

A global attribute index is an optional index and is referenced when the Directory Proxy Server is handling a search request with an equality filter involving the attribute, such as the `telephoneNumber` attribute with the filter (telephoneNumber=+11234567890). Since the Directory Proxy Server does not know what the data within the subtree views looks like or how it will be searched, it cannot create or recommend default global attribute index definitions. The creation of a global attribute index is based on the range of equality-filtered search requests that the Directory Proxy Server will handle. The Directory Server must also have an equality or ordering index type for the associated attribute Local DB Index."

The common candidates for global attribute indexing are the uniquely-valued equality-indexed attributes on the external servers. Examples of these attributes are `uid`, `mail` and `telephoneNumber`. Though the values of the attribute need not be unique to be used as a global attribute index by the entry-balancing request processor.

Consider a Directory Proxy Server deployment that expects to handle frequent searches of the form `"(&(mail=user@example.com)(objectclass=person))"`. Since the filter is constructed with an equality match and &-clause, we can use a global attribute index on the `mail` attribute to avoid forwarding the search request to each entry balanced dataset.

The following `dsconfig` command creates the global attribute index. Note that the mail attribute must be indexed for equality searches on each of the external servers behind the Directory Proxy Server.

```
$ bin/dsconfig create-global-attribute-index \
  --processor-name ou_people_dc_example_dc_com-eb-req-processor \
  --index-name mail --set prime-index:true \
```

After creating the index with `dsconfig`, the index will begin to be populated as search requests involving the mail attribute are made to the Directory Proxy Server. At this point, you can also use the `reload-index` tool to fully populate the index for optimal performance as described in the following section.

Reloading the Global Indexes

The Directory Proxy Server provides a tool, `reload-index`, which allows you to manually reload the Directory Proxy Server global indexes. You might need to reload the index when:

- The Directory Proxy Server fails to successfully load its global indexes on startup.
- Changes are made to the set of indexed attributes.
- Significant changes are made to the content in backend servers.
- The integrity of the index is in question.

You can use the tool to reload all configured indexes in the global index, including the RDN index and all attribute indexes, or to reload only those indexes you specify.

The tool schedules an operation to run within the Directory Proxy Server's process. You must supply LDAP connection information so that the tool can communicate with the server through its task interface. Tasks can be scheduled to run immediately or at a later scheduled time. Once scheduled, you can manage the tasks using the `manage-tasks` tool.

To Reload All of the Index

- Run the `reload-index` tool to reload all of the indexes within the scope of the `dc=example,dc=com` base DN. The task is performed as `cn=Directory Manager` on port 389 of the localhost server. The existing index contents are erased before reloading.

```
$ bin/reload-index --task --bindPassword password --baseDN
 "dc=example,dc=com"
```

To Reload the RDN and UID Index

- To reload the RDN and UID index in the background so that the existing contents of these indexes can continue to be used, run the command as follows:

```
$ bin/reload-index --task --bindPassword password \
  --baseDN "dc=example,dc=com" --index rdn --index uid --background
```

To Prime the Backend Server Using the --fromDS Option

You can force the Directory Proxy Server to prime from the backend directory server only using the --fromDS option, overriding the configuration of the `prime-index-source` property. You can do this on a one off basis if the global index appears to be growing too large. For example, run the command as follows:

- Run the `reload-index` command with the --fromDS option to prime the backend server.

```
$ bin/reload-index --bindPassword password --baseDN "dc=example,dc=com" --
fromDS
```

Monitoring the Size of the Global Indexes

Over time, stale entries can build up in the global indexes because proxies do not communicate changes to the indexes with one another. The Directory Proxy Server continues to operate normally in this situation since the global indexes are only ever used as a hint at where to find entries.

The rate of this growth is typically very slow since in most environments the key attributes change infrequently. The global indexes themselves are also very compact. However, if the global indexes start to fill up the allocated memory, you may need to flush and reload them. The size of the global indexes can be monitored over LDAP using the following command:

```
$ bin/ldapsearch -b "cn=monitor" -D "uid=admin,dc=example,dc=com" -w password
 \
  "(objectClass=ds-entry-balancing-request-processor-monitor-entry)" \
  global-index-current-memory-percent
```

If the global indexes fill up, the Directory Proxy Server will continue to operate normally, but it will need to start evicting entries from the indexes, which will lead to more broadcast searches, reducing the overall throughput of the Directory Proxy Server.

To reload the indexes so that they no longer hold stale information, run the `reload-index` command with the --fromDS option so that data is loaded from backend directory servers. We recommend that you reload the indexes during off-peak hours because it may have an impact on performance while the reload is in progress.

Sizing the Global Indexes

The Directory Proxy Server includes a tool, `global-index-size`, to help you estimate the size in memory of your global indexes. You can estimate the size of more than one index in a single invocation by providing multiple sets of options. The tool creates its estimate using the following information:

- **Number of keys in the index**. For example, for the built-in RDN index, the number of keys is the total number of entries in the Directory Server that are immediately below the balancing point. Entries more than one level below the balancing point, as well as entries that are not subordinate to the balancing point, will not be contained in the RDN index. For attribute indexes, the number of keys will be the number of unique values for that attribute in the entry-balanced portion of the data.
- **Average size of each key, in bytes**. For attributes indexes, the key is simply the attribute value. For the built-in RDN index, the key is the RDN directly below the balancing base DN. For example, for the DN `uid=user.0,dc=example,dc=com` under the balancing base DN of `dc=example,dc=com`, the key size is 10 bytes (the number of bytes in the RDN uid=user.0).
- **Estimated number of keys**. This value corresponds to the maximum number of keys you expect in your Directory Server. The number of keys is provided in the index-size configuration property of the `global-attribute-index` object when you configure an attribute index. For the built-in RDN index, the configured number of keys is provided in the `rdn-index-size` property. If you do not provide a value, the tool assumes that the configured number of keys is the same as the actual number of keys.

To Size the Global Index

- Run the `global-index-size` to estimate the size of two separate indexes, both with 10,000,000 keys but with differing average key sizes. The configured number of keys is assumed to be equal to the actual number of keys:

```
$ bin/global-index-size --numKeys 10000000 \
  --averageKeySize 11 --numKeys 10000000 \
  --averageKeySize 15

Num Keys : Cfg. Num Keys : Avg. Key Size : Est. Memory Size
---------:---------------:---------------:-----------------
10000000 : 10000000      : 11            : 159 mb
10000000 : 10000000      : 15            : 197 mb
```

Priming the Global Indexes on Start Up

The Directory Proxy Server can prime the global indexes on startup from the backend directory server or from a peer proxy server, preferably one that resides on the same LAN or subnet. When priming occurs locally, you can avoid WAN bandwidth consumption and reduce the processing load on the directory servers in the topology. You can specify the data sources for the index priming and the order in which priming from these sources occurs.

Use the `prime-index-source` property to specify the sources of data, either `ds`, `file` or some combination of the two. The order you specify is the order in which priming from these sources will be attempted. For example, if you specify `prime-index-source:file,ds`, priming will be performed from the `global-index` data file created from the previous run of the directory servers. With the `file,ds` configuration, the contents of the global index are written to disk periodically if, and only if, the entire global index has been primed previously from a directory servers source either from startup or reloaded-index. Priming is most efficient if the source server is on the same local network as the Directory Proxy Server.

To Configure All Indexes at Startup

The following example configures the entry-balancing request processor so that it primes the global index from the persisted file, if present, or from an external directory servers source if necessary.

- Run the `dsconfig` tool to prime all indexes at startup.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set prime-all-indexes:true --set prime-index-source:file \
  --set prime-index-source:ds
```

To Configure the Global Indexes Manually

If you do not want to configure priming during setup, you can configure index priming manually by creating an external server, creating a global attribute index, and then changing the entry-balancing request processor to load indexes from this external server.

1. Use the `dsconfig` tool to create an external server of the type PingDirectoryProxy Server to represent a peer of the Directory Proxy Server.

```
$ bin/dsconfig create-external-server \
  --server-name intra-proxy-host.example.com:3389 \
  --type PingDirectoryProxy-server \
  --set server-host-name:intra-proxy-host \
  --set server-port:338 \
  --set "bind-dn:cn=Directory Manager" \
  --set "password:secret123"
```

2. Create a global attribute index on the `uid` attribute as follows:

```
$ bin/dsconfig create-global-attribute-index \
  --processor-name dc_example+dc+com-eb-req-processor \
  --index-name uid \
```

3. Change the entry-balancing request processor to load the indexes at startup from the peer Directory Proxy Server using `dsconfig set-request-processor-prop` as described above.

To Persist the Global Index from a File

The PingDirectoryProxy Server supports periodically persisting the global index to a file and priming the global index from the persisted file when the server is restarted.

An Entry Balancing Request Processor can be configured to periodically persist the global index to disk, so that when the Entry Balancing Request Processor is reinitialized (on startup), it can prime the values from disk instead of putting load on the remote servers. Being able to read the index from disk eliminates the load on backend Directory Server instances if many PingDirectoryProxy Server instances were to come up at once.

An entry-balancing request processor can be configured to persist the global index to disk by including `file` as one of the prime index sources (with the `prime-index-source` property). The frequency at which the file is written is controlled by the `persist-global-index-frequency` property.

The global index needs to be fully primed before it will be persisted. It can be initially primed using a peer PingDirectoryProxy Server or from a backend Directory Server. On a running PingDirectoryProxy Server, when new global attribute indexes are added, the global index can be primed with those attribute indexes by running the `rebuild-index` tool. The `rebuild-index` tool always uses a remote server for priming the global index even if `file` is configured as a source). On subsequent restarts of the PingDirectoryProxy Server, the global index will be primed from the persisted file instead of going over the network to a remote server, which allows it to be primed much faster than if it were using a remote priming source. Also, during server startup, the global index priming works by using each configured `prime-index-source` property in the specified order until it is fully primed to take advantage of what is available locally before contacting one or more remote servers.

- The following `dsconfig` command prime all indexes at startup from a file.

```
dsconfig -n set-request-processor-prop \
  --processor-name entry-balancing \
  --set prime-index-source:file \
  --set prime-index-source:ds \
  --set persist-global-index-frequency:10s \
  --set persist-global-index-directory:/servers/proxy-1/index-files \
  --set prime-all-indexes:true
```

Priming or Reloading the Global Indexes from Sun Directory Servers

When priming or reloading a global index based on a Sun Directory Server environment, the Sun servers may become overwhelmed and unresponsive because of their method of streaming data. To reduce the impact of priming on these server, you can use the `prime-search-entry-per-second` property. To reduce the impact of reloading these indexes, use the `--searchEntryPerSecond` property of the `reload-index` command. These properties control the rate at which the Directory Proxy Server accepts search result entries from the backend directory servers.

To find the optimum rate, start low and specify a few thousand search entries per second. Then increase as necessary.

Working with Alternate Authorization Identities

Access control rules in an entry-balanced deployment are configured in the Directory Server backend servers and require access to the entry contents of the user *issuing* the request. This can introduce a possible issue when clients to the Directory Proxy Server authenticate as users whose entries are among the entry-balanced sets. If the server which is processing a request does not contain the issuing user's entry, then the access control cannot be evaluated.

For example, consider a deployment that has two entry-balancing sets, set-01 and set-02. Set-01 has entries in the range `uid=0-10000`, while set-02 has entries for `uid=10001-20000`. The client with `uid=5000` binds to the Directory Proxy Server, which sends a BIND request to entry-balancing set-01. Next, the client sends a SEARCH request with filter "`(uid=15000)`". The Directory Proxy Server determines that `uid=15000` lives on entry-balancing set-02. The Directory Proxy Server then determines that the entry for the authenticated user with `uid=5000` does not exist in set-02 and that the access control handler would reject the SEARCH request issued by an unknown user.

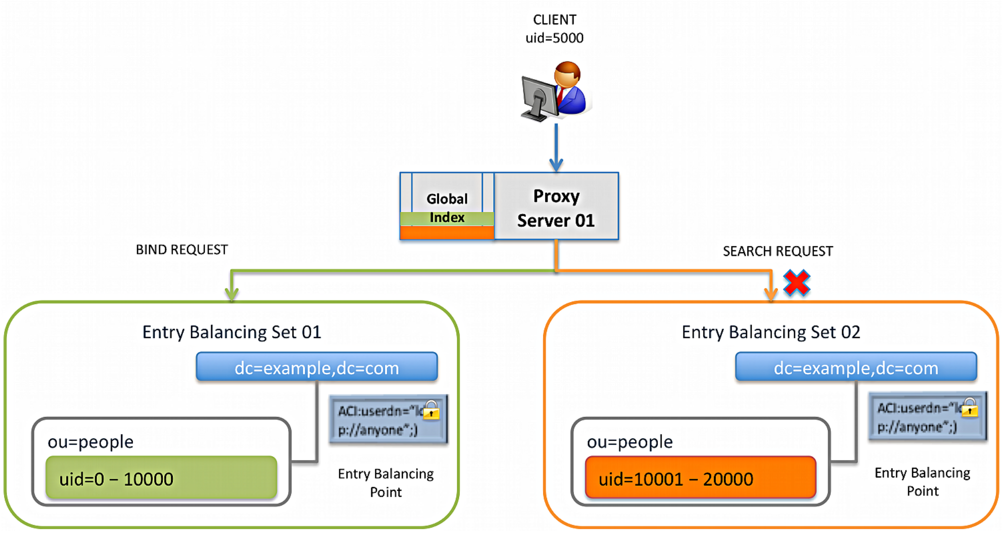


Figure 5: Entry-Balancing Issue with Clients Not Present in the Underlying Data Set

One solution to this problem is to make use of an *alternate authorization identity* for the user, which references an entry that exists in all Directory Servers in all backend sets and has an equivalent set of access control rights as the authenticated user. The alternate authorization identity is used when the Directory Proxy Server observes that the Directory Server processing a request does not contain the entry of the user issuing the request.

The following sections cover the procedures to configure the alternate authorization identities for the Directory Proxy Server.

About Alternate Authorization Identities

Whenever the Directory Proxy Server forwards a request to the backend set containing the user's entry, it forwards the request with an authorization identity that reflects the user's actual identity, since servers in that set already know about that user. However, when forwarding a request to a backend set that does *not* contain the user's entry, the Directory Proxy Server uses an *alternate authorization identity* that reflects the generic user with the same set of rights as the actual user issuing the request. Alternate authorization identities allow for the proper evaluation of access control rules for users whose entries are not present within an entry-balanced dataset.

There are typically only a few different generic class of users from an access control perspective, which can be placed in a portion of the DIT that is not below the entry-balancing base DN and is replicated to all servers in the topology. For example, assume that you have three classes of users: full administrators, password administrators, and normal users. You could create the following entries in the topology and assign them the appropriate access rights:

uid=normal user,dc=example,dc=com
uid=server-admin,dc=example,dc=com
uid=password-admin,dc=example,dc=com

Returning to the example scenario, the client with `uid=5000` binds to the Directory Proxy Server, which sends a BIND request to entry-balancing set-01. Next, the client sends a SEARCH request for `uid=15000`. The Directory Proxy Server determines that `uid=15000` lives on entry-balancing set-02. Next, the Directory Proxy Server then determines that the client `uid=5000` does not have an entry on entry-balancing set-02. The Directory Proxy Server uses an *alternate authorization identity* that reflects the generic user, `uid=normal user`, which has the same set of rights as the client `uid=5000` who is issuing the request. The access control is accepted and the SEARCH request returns a response for `uid=5000`.

Whenever a user authenticates to the Directory Proxy Server, the server can keep track of which backend set holds that user's entry and determine whether an alternate authorization identity is required. The server can also determine which of these generic accounts best describes the rights that the user should have.

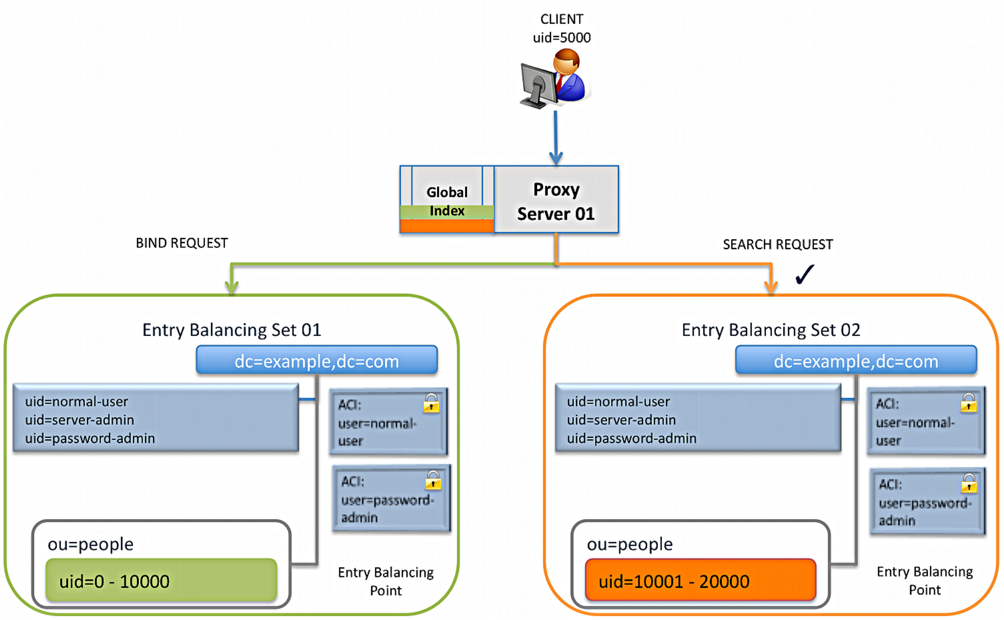


Figure 6: Alternate Authorization Identity Solves Access Control Issues in Entry-Balancing Deployments

When an alternate authorization identity is invoked, you will see `authzID='dn:uid=normal user,dc=example,dc=com'` in the server log, indicating that the alternate authorization identity was used. For example, if the `user.15000` is in a different backend set from `user.5000`, the log will show the following:

```
% bin/ldapsearch -D "uid=user.5000,ou=people,dc=example,dc=com" -w pasword \
  -b uid=user15000,ou=people,dc=example,dc=com "(objectclass=*)"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2
via="app='Directory-Proxy address='127.0.0.1'
authzID='dn:uid=normal user,dc=example,dcom' sessionID='conn=2'
requestID='op=1'" base="uid=user.150000,ou=people,dc=example,dc=com"scope=2
filter="(objectclass=*)" attrs="ALL"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2 resultCode=0
 etime=2.038
entriesReturned=1 authzDN="uid=normal-user,dc=example,dc=com"
```

Configuring Alternate Authorization Identities

Alternate authorization identities are specified by the `authz-attribute` property of the entry-balancing request processor configuration object. By default, the `authz-attribute` property has the default value of `ds-authz-map-to-dn`, which is an attribute reserved for this purpose.

To Configure Alternate Authorization Identity DNs

If a user entry has a value for `ds-authz-map-to-dn` whether it's explicitly contained in the entry or only present via a virtual attribute, then that will be used to specify the alternate authorization identity for the user. Otherwise, the default authorization identity (as indicated via the `authz-dn` configuration property) will be used to determine the alternate authorization identity.

1. Use `dsconfig` to set the `authz-dn` property of the entry-balancing request processor configuration. If any user among the balanced entries does not have an alternate authorization identity defined, the Directory Proxy Server will use the value of the `authz-dn` property of the entry-balancing request processor configuration.

   ```
   $ bin/dsconfig set-request-processor-prop \
     --processor-name dc_example_dc_com-eb-req-processor \
   ```

```
  --set "authz-dn:uid=normal user,dc=example,dc=com"
```

2. Create an auxiliary object class containing `ds-authz-map-to-dn` as an allowed attribute.
3. Add the auxiliary object class value to all user entries of interest.
4. Then, add the following attribute value to a `server-admin` user.

   ```
   ds-authz-map-to-dn: uid=server-admin,dc=example,dc=com
   ```

Chapter 7

Managing Entry-Balancing Replication

Topics:

Replication in the PingDirectoryProxy Server synchronizes directory data between all servers in the topology. In a deployment using the entry-balancing feature, however, directory data under the entry-balancing point is split into multiple data sets. Each data set is replicated to ensure high availability between a subset of the servers in the topology. Other directory data, such as the schema or data above the entry-balancing point, is replicated between all servers in the topology.

This chapter presents the following information about replication in an entry-balancing environment:

Overview of Replication in an Entry-Balancing Environment

In an entry-balanced deployment, some data is replicated everywhere, such as the schema, the server registry, and other shared data, and some data is replicated only on certain servers. A replication domain contains all of the servers in a replicated topology and shares a schema. The replication domain is associated with the base DN and must be a base DN of a backend.

By default, replication propagates updates to all replication servers in the topology. Updates to data under the entry-balancing point, however, must be replicated only among server instances in the same data set. Replication requires that, in such deployments, the Directory Server is configured with a replication set name global configuration property, and two backends. One backend has a base DN that is replicated globally (such as `dc=example,dc=com`) and the second backend has a base DN associated with the entry-balancing point (such as `ou=people,dc=example,dc=com`).

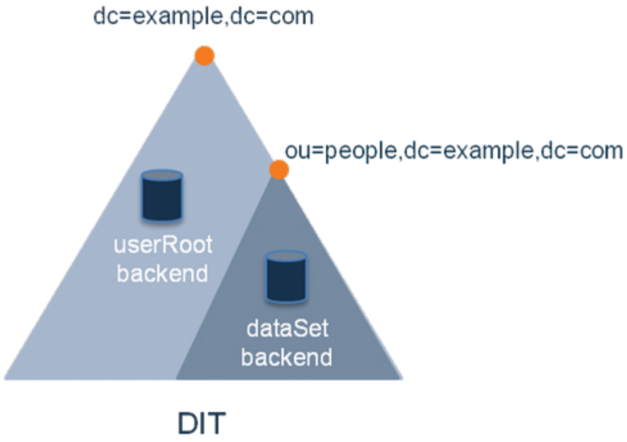


Figure 7: Global and Restricted Backends

If a data set name is not defined when you set up the Directory Proxy Server, one will be provided by default. The proper configuration of an entry-balancing environment requires coordination between the Directory Server and Directory Proxy Server. Once replication is enabled, the replication domain may be designated as the domain participating in entry balancing.

Review the *P Administration Guide* for more details about replication, managing the replication topology, and working with multiple backends.

Replication Prerequisites in an Entry-Balancing Deployment

Replication in an entry-balanced deployment requires the following:

- **Multiple local DB backends**. When you set up the Directory Server instances, you need two backends, a global backend for globally replicated data, such as `userRoot`, and a backend for the balancing point base DN, `dataSet`. Both backends need to be enabled for replication and initialized separately.
- **Replication set name**. Every Directory Server in your replicated topology must have a replication set name. This replication set name coordinates the Directory Proxy Server and the Directory Server. The restricted domain is only replicated within instances using the same replication set name.
- **Multiple Directory Proxy Server subtree views**. The entry-balanced proxy configuration relies on multiple subtree views, one for the globally replicated base DN and one for the entry-balancing point base DN. The globally replicated base DN will have a proxying request processor associated with it. The restricted base DN will have an entry-balancing request processor associated with it. This configuration is best achieved using the `create-initial-proxy-config` tool after running setup.

About the --restricted Argument of the dsreplication Command-Line Tool

When enabling replication for a server that takes part in an entry balanced environment, it is recommended that the multiple domains involved are enabled at the same time. There is a global domain, and a restricted domain, where the restricted domain represents the entry-balancing point. Each base DN is defined in a separate Local DB Backend. The `dsreplication` CLI tool has a --restricted argument that is used to specify which base DN is considered an entry-balancing point.

To Use the --restricted Argument of the dsreplication Command-Line Tool

- Run `dsreplication` to enable replication between two servers with entry balancing.
  - You can run the command in non-interactive mode as follows:

    ```
    $ bin/dsreplication enable --host1 host1.example.com \
      --port1 1389 --bindDN1 "cn=Directory Manager" \
      --bindPassword1 secret --replicationPort1 8989 \
      --host2 host2.example.com --port2 2389 \
      --bindDN2 "cn=Directory Manager" --bindPassword2 secret \
      --replicationPort2 8989 --baseDN dc=example,dc=com \
      --baseDN ou=people,dc=example,dc=com \
      --restricted ou=people,dc=example,dc=com
    ```
  - Alternatively, you can enable replication using the interactive command line, making sure to specify that an entry balancing is being used and specifying the base DN of the entry-balancing point. After entering `dsreplication` and entering the LDAP connection parameters, follow the prompts presented.

    ```
    You must choose at least one base DN to be replicated.

    Replicate base DN dc=example,dc=com? (yes / no) [yes]: yes

    Replicate base DN ou=people,dc=example,dc=com? (yes / no) [yes]: yes

    Do you plan to configure entry balancing using the Directory Proxy Server?
     (yes / no) [no]: yes

    Is dc=example,dc=com an entry-balancing point? (yes / no) [no]: no

    Is ou=people,dc=example,dc=com an entry-balancing point? (yes / no) [no]:
     yes
    ```

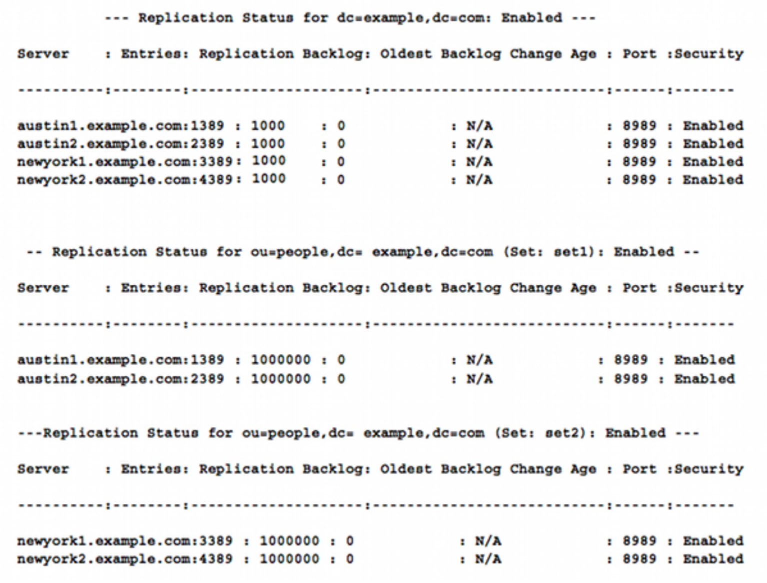
Checking the Status of Replication in an Entry-Balancing Deployment

You can use the `dsreplication` status tool to check the status of an entry-balancing deployment. In this example, the `ou=people,dc=example,dc=com` subtree is entry-balanced. The data is split into two sets, `set1` and `set2`. The servers `host1` and `host2` are in replication set `set1` and servers `host3` and `host4` are in replication set `set2`.

To Check the Status of Replication in an Entry-Balancing Deployment

- Run the `dsreplication` command to get a status of replication in the entry-balancing deployment. To view a specific set, use the --setName option to see only the specific replication set; otherwise, all of the sets will be displayed by default.

  ```
  $ bin/dsreplication status --hostname host1.example.com \
    --port 1389 --adminUID admin --adminPassword secret
  ```



```
          --- Replication Status for dc=example,dc=com: Enabled ---

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:-------------------:--------------------------:------:-------

austin1.example.com:1389 : 1000    : 0            : N/A            : 8989 : Enabled
austin2.example.com:2389 : 1000    : 0            : N/A            : 8989 : Enabled
newyork1.example.com:3389: 1000    : 0            : N/A            : 8989 : Enabled
newyork2.example.com:4389: 1000    : 0            : N/A            : 8989 : Enabled


 -- Replication Status for ou=people,dc= example,dc=com (Set: set1): Enabled --

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:-------------------:--------------------------:------:-------

austin1.example.com:1389 : 1000000 : 0            : N/A           : 8989 : Enabled
austin2.example.com:2389 : 1000000 : 0            : N/A           : 8989 : Enabled

---Replication Status for ou=people,dc= example,dc=com (Set: set2): Enabled ---

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:-------------------:--------------------------:------:-------

newyork1.example.com:3389 : 1000000 : 0            : N/A            : 8989 : Enabled
newyork2.example.com:4389 : 1000000 : 0            : N/A            : 8989 : Enabled
```

Example of Configuring Entry-Balancing Replication

This section describes how to set up a four-server replication topology that uses entry balancing to distribute entries across the servers. The procedure assumes that none of the servers have participated in any previous replication topology. This is supported for one or multiple entry balancing domains.

Assumptions

The example uses the LDAP (389) and replication (8989) ports respectively. It configures the following hosts:

austin1.example.com
newyork1.example.com
austin2.example.com
newyork2.example.com

In this example, we have a global domain of `dc=example,dc=com`, which is replicated across all servers. The data below the entry-balancing point of `ou=people,dc=example,dc=com` is distributed across two data sets, dataSet1 and dataSet2. Each data set is replicated between two directory servers. Each of these servers is associated with one of two locations, Austin and New York.

Configuration Summary

To configure replication in an entry-balanced deployment, you must do the following:

- Install two directory servers in an Austin location and two directory servers in a New York location.
- Create a new backend, called `dataset`, to store the entry-balancing data set.
- Define entry-balancing set names `dataSet1` and `dataSet2` for assignment to the `replication-set-name` Global Configuration Property of the Directory Server instances.
- Import the data representing the global domain, stored in userRoot, into a server. Choose a server for each of the entry-balancing data sets, both stored in the backend named `dataset`.
- Enable replication and initialize remaining servers.
- Configure the proxies.
- Check the status of replication.

To Install the Directory Server

First, install the Directory Server instances. In this example, we install the following four servers, two in the Austin location and two in the New York location:

austin1.example.com
austin2.example.com
newyork1.example.com
newyork2.example.com

1. We install the first server, austin1, as follows:

```
root@austin1# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

2. Install the second Austin server, austin2, in the same way:

```
root@austin2 # ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

3. Next, install the two New York servers, newyork1 and newyork2, as follows:

```
root@newyork1# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense

root@newyork# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

To Create the Database Backends and Define the Replication Set Name

1. On all servers, create the dataset backend as follows:

```
./bin/dsconfig --no-prompt create-backend \
--backend-name dataset --type local-db --set enabled:true \
--set base-dn:ou=people,dc=example,dc=com
```

2. Set the replication set name for `austin1.example.com` and `newyork1.example.com` to dataset1:

```
./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset1
```

3. Set the replication set name for `austin2.example.com` and `newyork1.example.com`to dataset2:

```
./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset2
```

To Create and Set the Locations

1. On the Austin servers, create the two locations, newyork and austin, and set the location of this instance to austin:

```
./bin/dsconfig --no-prompt create-location --location-name austin

./bin/dsconfig --no-prompt create-location --location-name newyork \
--set preferred-failover-location:austin

./bin/dsconfig --no-prompt set-location-prop --location-name austin \
--add preferred-failover-location:newyork

./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:austin
```

2. For the New York servers, set the location to newyork:

```
./bin/dsconfig --no-prompt create-location \
--location-name austin

./bin/dsconfig --no-prompt create-location \
--location-name newyork \
--set preferred-failover-location:austin

./bin/dsconfig --no-prompt set-location-prop \
--location-name austin \
--add preferred-failover-location:newyork

./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:newyork
```

To Import the Entries

We import the userRoot data, based on data defined in the `userRoot.ldif` file, into one server. This file does not contain entries at or within the entry-balancing point, `ou=people,dc=example,dc=com`.

1. Use the `import-ldif` command to import the userRoot data.

```
root@austin1# ./bin/import-ldif --backendID userRoot \
--ldifFile /data/userRoot.ldif \
--includeBranch dc=example,dc=com \
--rejectFile /data/austin1-import-rejects \
--port 389
--hostname austin1.example.com
```

2. Import the dataSet1 data on one server into the dataset backend, which is assigned the dataSet1 `replication-set-name`.

```
root@austin1# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset1.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/austin1-dataset-import-rejects \
--hostname austin1.example.com --port 389
```

3. Import the dataSet2 data on one server into the dataset backend, which is assigned the dataSet2 `replication-set-name`.

```
root@austin2# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset2.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/austin2-dataset-import-rejects \
--hostname austin2.example.com --port 389
```

To Enable Replication in an Entry-Balancing Deployment

Now we can enable replication between the servers and initialize the remaining servers without data. Notice that we specify the `--restricted` domain in the `dsreplication` command.

1. Run `dsreplication enable` to enable the servers in the topology. The first invocation of this command creates the admin account.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 austin2.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

2. Enable replication between austin1 and newyork1. This procedure automatically enables replication between austin2 and newyork1 as well.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 newyork1.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

3. Enable replication between austin1 and newyork2. This will complete the entry-balancing replication setup.

```
root@austin1# ./bin/dsreplication enable \
  --host1 austin1.example.com \
  --port1 389 --bindDN1 "cn=directory manager" \
  --bindPassword1 pass --host2 newyork2.example.com \
  --port2 389 --bindDN2 "cn=directory manager" \
  --bindPassword2 pass \
  --replicationPort1 8989 \
  --replicationPort2 8989 \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --restricted ou=people,dc=example,dc=com \
  --adminUID admin --adminPassword pass --trustAll \
  --no-prompt
```

4. Initialize the remaining servers without data. The global domain, `dc=example,dc=com` needs to be initialized on austin2, newyork1 and newyork2. The `ou=people,dc=example,dc=com` entry-balancing domain needs to be initialized from austin1 to newyork2, and then again from austin2 to newyork2. We will combine these steps by initializing both domains with one invocation once austin2 is initialized with the global domain.

```
root@austin1# ./bin/dsreplication initialize \
  --hostSource austin1.example.com --portSource 389 \
  --hostDestination austin2.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --no-prompt

root@austin1# ./bin/dsreplication initialize \
  --hostSource austin1.example.com --portSource 389 \
  --hostDestination newyork1.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --no-prompt

root@austin2# ./bin/dsreplication initialize \
  --hostSource austin2.example.com --portSource 389 \
  --hostDestination newyork2.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --no-prompt
```

To Check the Status of Replication

Once replication has been configured, check the status of the replication topology using the `dsreplication status` command.

- Run the `dsreplication status` command to check its status.

```
root@austin1# ./bin/dsreplication status \
--adminPassword pass --no-prompt --port 389
```

Chapter 8

Managing the Directory Proxy Server

Topics:

Once you have configured the PingDirectoryProxy Server, you can manage the day-to-day operations of your deployment using the monitoring and logging features. This chapter provides procedures to help you configure logging and monitor your deployment.

This chapter includes the following sections:

Managing Logs

The Directory Proxy Server provides a number of different types of log publishers that can be used to provide information about how the server is processing.

About the Default Logs

You can view all logs in the `PingDirectoryProxy/logs` directory. This section provides information about the following default logs:

Error Log
server.out Log
Debug Log
Config Audit Log and the Configuration Archive
Access Log
Setup Log
Tool Log
LDAP SDK Debug Log

Error Log

By default, this log file is available at `logs/errors` below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE msgID=458887
 msg="The Directory Server has started successfully"
```

The category field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL, ADMIN, ADMIN_TOOL, BACKEND, CONFIG, CORE, DSCONFIG, EXTENSIONS, PROTOCOL, SCHEMA, JEB, SYNC, LOG, PLUGIN, PROXY, QUICKSETUP, REPLICATION, RUNTIME_INFORMATION, TASK, THIRD_PARTY, TOOLS, USER_DEFINED, UTIL, VERSION.

The severity field provides information about how severe the server considers the problem to be. Available severities include:

- **DEBUG** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Proxy Server provides a separate debug logging facility as described below.
- **INFORMATION** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.
- **MILD_WARNING** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.
- **MILD_ERROR** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.
- **NOTICE** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.
- **SEVERE_WARNING** – Used for problems that the server detects that might lead to bigger problems in the future and should be addressed by administrators.
- **SEVERE_ERROR** – Used for significant problems that have prevented the server from successfully completing processing and are considered important.

- **FATAL_ERROR** – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the NOTICE, SEVERE_WARNING, SEVERE_ERROR, and FATAL_ERROR severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the PROTOCOL category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-
warning,severe-error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the DEBUG severity for a category will not automatically enable logging at the INFORMATION, MILD_WARNING, or MILD_ERROR severities.

The error log configuration may be altered on the fly using tools like `dsconfig`, the Administrative Console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out Log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. The `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug Log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- **enabled** – Indicates whether debug logging is enabled. By default, it is disabled.
- **log-file** – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- **default-debug-level** – Specifies the minimum log level for debug messages that should be written. The default value is “error,” which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include warning, info, and verbose. Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the info debug level automatically enables the warning and error levels.
- **default-debug-category** – Specifies the categories for debug messages that should be written. Some of the most useful categories include caught (provides information and stack traces for any exceptions caught during processing), database-access (provides information about operations performed in the underlying database), protocol (provides information about ASN.1 and LDAP communication performed by the server), and data (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files to help isolate information that might be relevant to a particular problem.

Note: Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and then be scoped so that only pertinent debug information is recorded.

Debug targets can be used to further pare down the set of messages generated. For example, you can specify that the debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and interpret the output.

Audit log

The audit log is a specialized version of the access log, used for troubleshooting problems that may occur in the course of processing. The log records all changes to directory data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

By default, if you enable the audit log on the Directory Proxy Server, the `userPassword` and `authPassword` attribute values are obscured, Each value of an obscured attribute is replaced in the audit log with a string of the form `"***** OBSCURED VALUE *****"`. You can unobscure these attributes by deleting them from the `obscure-attribute` property.

Config Audit Log and the Configuration Archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the `logs/config-audit.log` file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the `dsconfig` command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.
- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Proxy Server maintains complete copies of all previous configurations. These configurations are provided in the `config/archived-configs` directory and are gzip-compressed copies of the `config/config.ldif` file in use before the configuration change was made. The filenames contain time stamps that indicate when that configuration was first used.

Access and Audit Log

The access log provides information about operations processed within the server. The default access log file is written to `logs/access`, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1"
 to="127.0.0.1"
  protocol="LDAP+TLS" clientConnectionPolicy="default"
```

```
[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
  dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
 clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2
  base="ou=People,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL"
  resultCode=0 etime=0.549 entriesReturned=1
[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensi- tive or not useful.

Result messages include a `resultCode` element that indicates whether the operation was successful or if failed and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **origin=replication** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **message** – Text that was included in the `diagnosticMessage` field of the response sent to the client.
- **additionalInfo** – Additional information about the operation that was not included in the response sent back to the client.
- **authDN – DN** of the user that authenticated to the server (typically only included in bind result messages).
- **authzDN – DN** of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **authFailureID** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).
- **authFailureReason** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **responseOID** – OID included in an extended response returned to the client.
- **entriesReturned** – Number of matching entries returned to the client for a search operation.
- **unindexed=true** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The LDAP SDK for Java provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Proxy Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify DN) processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to `logs/audit`) in LDIF form.

The PingDirectoryProxy Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup Log

The `setup` tool writes a log file providing information about the processing that it performs. By default, this log file is written to `logs/setup.log` although a different name may be used if a file with that name already exists, because the `setup` tool has already been run. The full path to the setup log file is provided when the `setup` tool has completed.

Tool Log

Many of the administrative tools provided with the Directory Proxy Server (for example, `import-ldif`, `export-ldif`, `backup`, `restore`, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the `logs/tools` directory (for example, **logs/tools/ import-ldif.log**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

LDAP SDK Debug Log

This log can be used to help examine the communication between the Directory Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

Types of Log Publishers

The PingDirectoryProxy Server provides a number of differently types of loggers that can be used to get processing information about the server. There are three primary types of loggers:

- **Access loggers** provide information about operations processed within the server. They can be used for understanding the operations performed by clients and debugging problems with directory-enabled applications, and they can also be used for collecting usage information for performance and capacity planning purposes.
- **Error loggers** provide information about warnings, errors, or significant events that occur within the server.
- **Debug loggers** can provide detailed information about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database.

By default, the following log publishers are enabled on the system:

File-based access logger
File-based error logger
Failed-operations access logger

The PingDirectoryProxy Server also provides the follow log publishers that are disabled by default:

File-based debug logger
File-based audit logger
Expensive operations access logger
Successful searches with no entries returned access logger

Creating New Log Publishers

The PingDirectoryProxy Server provides customization options to help you create your own log publishers with the `dsconfig` command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see Configuring Log Rotation and Configuring Log Retention.

To Create a New Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

Note: To configure compression on the logger, add the option to the previous command:

```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. If needed, view log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. On the main menu, select the option to configure the log publisher.
4. On the **Log Publisher** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the Directory Proxy Server root. For example, `logs/disconnect.log`.
9. Select the rotation policy to use for this log publisher.
10. Select the retention policy to use for this log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Type `f` to apply the changes.

About Log Compression

The Directory Proxy Server supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The `summarize-access-log` tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting purposes, administrators using compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the `compression-mechanism` property to have the value of "gzip" when creating a new logger.

About Log Signing

The Directory Proxy Server supports the ability to cryptographically sign a log to ensure that it has not been modified in any way. For example, financial institutions require audit logs for all transactions to check for correctness. Tamper-proof files are therefore needed to ensure that these transactions can be propertly validated and ensure that they have not been modified by any third-party entity or internally by unscrupulous employees. You can use the `dsconfig` tool to enable the `sign-log` property on a Log Publisher to turn on cryptographic signing.

When enabling signing for a logger that already exists and was enabled without signing, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the `validate-file-signature` tool provided in the `bin` directory of the server (or the `bat` directory for Windows systems).

Once you have enabled this property, you must disable and then re-enable the Log Publisher for the changes to take effect.

About Encrypting Log Files

The Directory Proxy Server supports the ability to encrypt log files as they are written. The `encrypt-log` configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an `.encrypted` extension (and if both encryption and compression are enabled, the extension will be `.gz.encrypted`). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The `encrypt-file` tool (or custom code, using the LDAP SDK's `com.unboundid.util.PassphraseEncryptedInputStream`) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the `encryption-settings create` command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the `encryption-settings-definition-id` property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the `rotate-log` tool to force the file to be rotated before attempting to examine it.

To Configure Log Signing

1. Use `dsconfig` to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set enabled:false
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set enabled:true
```

To Validate a Signed File

The Directory Proxy Server provides a tool, `validate-file-signature`, that checks if a file has not been tampered with in any way.

- Run the `validate-file-signature` tool to check if a signed file has been tampered with. For this example, assume that the `sign-log` property was enabled for the File-Based Audit Log Publisher.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

Note: If any validations errors occur, you will see a message similar to the one as follows:

```
One or more signature validation errors were encountered
while validating the contents of file 'logs/audit':
* The end of the input stream was encountered without
  encountering the end of an active signature block.
  The contents of this signed block cannot be trusted
  because the signature cannot be verified
```

To Configure Log File Encryption

1. Use `dsconfig` to enable encryption for a Log Publisher. In this example, the File-based Access Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted Access"
 \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-definition-
id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...Done
```

```
Writing decrypted data to file '/ds/PingDirectoryProxy/decrypted-access'
 using a
key generated from encryption settings definition
 '332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'
Successfully wrote 123,456,789 bytes of decrypted data
```

Configuring Log Rotation

The Directory Proxy Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Proxy Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

To Configure the Log Rotation Policy

- Use `dsconfig` to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring Log Rotation Listeners

The Directory Proxy Server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed. Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true
```

The path specified by the `copy-to-directory` property must exist, and the filesystem containing that directory must have enough space to hold all of the log files that will be written there. The server will automatically monitor free disk space on the target filesystem and will generate administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the `summarize-access-log` tool on a recently-rotated log file and writes its output to a file in a specified location. This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the `summarize-access-log` tool, including the `file-`

`based-access` and `operation-timing-access` logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of `.summary`. If the `output-directory` property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically be deleted. Though files are generally small in comparison to the log files themselves, make sure that there is enough space available in the specified storage directory. The server automatically monitors free disk space on the filesystem to which the summary files are written.

Configuring Log Retention

The Directory Proxy Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Proxy Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Directory Proxy Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Custom Retention Policy**. Create a new retention policy that meets your Directory Proxy Server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

- Use `dsconfig` to modify the log retention policy for the access logger.

  ```
  $ bin/dsconfig set-log-publisher-prop \
    --publisher-name "File-Based Access Logger" \
    --set "retention-policy:Free Disk Space Retention Policy"
  ```

Setting Resource Limits

You can set resource limits for the Directory Proxy Server using several global configuration properties as well as setting resource limits on specific client connection policies. If you configure both global and client connection policy resource limits, the first limit reached will always be honored. For example, if the server-wide maximum concurrent connections limit is reached, then all subsequent connection will be rejected until existing connections are closed, regardless of whether a client connection policy limit has been reached.

Setting Global Resource Limits

You can specify the following types of global resource limits:

- Specify the maximum number of client connections that can be established at any given time using the `maximum-concurrent-connections` property. If the server already has the maximum number of connections established, then any new connection attempts from any clients will be rejected until an existing connection is closed. The default value of zero indicates that no limit is enforced.
- Specify the maximum number of client connections that can be established at any give time from the same client system using the `maximum-concurrent-connections-per-ip-address` property. If the server already has the maximum number of connections established from a given client, then any new connection attempts from that client will be rejected until an existing connection from that client is closed. The server may continue to accept connections from other clients that have not yet reached this limit. The default value of zero indicates that no limit is enforced.
- Specify the maximum number of client connections that can be established at any given time while authenticated as a particular user with the `maximum-concurrent-connections-per-bind-dn` property. This property applies after the connection is established, because the bind operation to authenticate the user happens after the connection is established rather than during the course of establishing the connection itself. If the maximum number of connections are authenticated as a given user, then any new attempt to authenticate as that user will cause the connection performing the bind to be terminated. Note that this limit applies only to authenticated connections, and will not be enforced for clients that have not authenticated or for clients that have authenticated as the anonymous user. The default value of zero indicates that no limit is enforced.

Any changes to the `maximum-concurrent-connections` and `maximum-concurrent-connections-per-ip-address` properties will take effect only for new connections established after the change is made. Any change to the `maximum-concurrent-connections-per-bind-dn` property will apply only to connections (including existing connections) which perform authentication after the change is made. Existing connections will be allowed to remain established even if that would cause the new limit to be exceeded.

Setting Client Connection Policy Resource Limits

You can also configure resource limits in a client connection policy using the following properties of the client connection policy:

- **maximum-concurrent-connections**. This property specifies the maximum number of client connections that may be associated with a specific client connection policy at any given time. Once this limit has been reached, any further attempts to associate a connection with this client connection policy will result in the termination of the connection.
- **maximum-connection-duration**. This property specifies the maximum length of time that a connection associated with a particular client connection policy may be established. When the connection has been established longer than this period, it will be terminated.
- **maximum-idle-connection-duration**. This property specifies the maximum time that a connection associated with a particular client connection policy may remain established after the completion of the last operation processed on that connection. Any new operation requested on the connection resets the timer. Connections that are idle for longer than the specified time will be terminated.
- **maximum-operation-count-per-connection**. This property specifies the maximum number of operations that may be requested by any client connection associated with this client connection policy. If an attempt is made to process more than this number of operations on the connection, then the connection will be terminated.
- **maximum-concurrent-operations-per-connection**. This property specifies the maximum number of concurrent operations that can be in progress for any connection. This property can be used to prevent a single client connection from monopolizing server processing resources by sending a large number of concurrent asynchronous requests.
- **maximum-connection-operation-rate**. This property specifies the maximum rate at which a client associated with a specific client connection policy may issue requests to the Directory Proxy Server. If a client attempts to request operations at a rate higher than this limit, then the server will behave as described by the `connection-operation-rate-exceeded-behavior` property.
- **connection-operation-rate-exceeded-behavior**. This property describes how the server should behave if a client connection attempts to exceed a rate defined in the `maximum-connection-operation-rate` property.
- **maximum-policy-operation-rate**. This property specifies the maximum rate at which all clients associated with a particular client connection policy may issue requests to the Directory Proxy Server. If this limit is exceeded, then

the server will exhibit the behavior described in the `policy-operation-rate-exceeded-behavior` property.

- **policy-operation-rate-exceeded-behavior**. This property specifies the behavior of the Directory Proxy Server if a client connection attempts to exceed the rate defined in the `maximum-policy-operation-rate` property.

Monitoring the Directory Proxy Server

While the Directory Proxy Server is running, it generates a significant amount of information available through monitor entries. This section contains information about the following:

- Monitoring Server Status Using the status Tool
- About the Monitor Entries
- Using the Monitoring Interfaces
- Monitoring with JMX

Monitoring System Data Using the PingDataMetrics Server

The PingDataMetrics Server provides collection and storage of performance data from your server topology. You can use the System Utilization Monitor with the PingDataMetrics Server to collect information about the host system CPU, disk, and network utilization on any platform except Linux. If you are not using the PingDataMetrics Server, you do not need to use the system utilization monitor. When data is being collected, it periodically forks the process and executes commands.

For more information about using the System Utilization Monitor, refer to the data collection chapter of the PingDataMetrics Server documentation.

To Monitor Server Using the Status Tool

The PingDirectoryProxy Server provides a `status` tool that provides basic server status information, including version, connection handlers, a table of LDAP external servers, and the percent of the global index that is used.

1. Run the `status` tool to view the current state of the server.

```
$ bin/status
```

2. Enter the LDAP connection parameters.

```
>>>> Specify LDAP connection parameters

Administrator user bind DN [cn=Directory Manager]:

Password for user ‘cn=Directory Manager’:
```

```
          --- Server Status ---
Server Run Status:    Started 07/Jan/2011:10:59:52.000 -0600
Operational Status:   Available
Open Connections:     4
Max Connections:      8
Total Connections:    25

          --- Server Details ---
Host Name:            example
Administrative Users: cn=Directory Manager
Installation Path:    /path/to/PingDirectoryProxy
Version:              PingDirectoryProxy Server 7.0.1.0
Java Version:         jdk-7u9

          --- Connection Handlers ---

Address:Port : Protocol : State
```

```
-------------:----------:---------

0.0.0.0:1689 : JMX      : Disabled
0.0.0.0:636  : LDAPS    : Disabled
0.0.0.0:9389 : LDAP     : Enabled

         --- LDAP External Servers ---

Server         : Status    : Score : LB Algorithm
---------------:------------:-------:----------------------
localhost:389  : Available : 10    : dc_example_dc_com-failover
localhost:1389 : Available : 10    : dc_example_dc_com-failover


         --- LDAP External Server Op Counts ---

Server         : Add : Bind:Compare:Delete:Modify:Mod DN:Search : All
---------------:-----:----:------:-------:-------:------:-------:----
localhost:11389: 0   : 0  : 0    : 0     : 0     : 0    : 1249  : 1249
localhost:12389: 0   : 0  : 0    : 0     : 0     : 0    : 494   : 494



         --- Entry Balancing Request Processors ---

 Base DN                     : Global Index % Used
----------------------------:--------------------
ou=people,dc=example,dc=com : 33


         --- Global Index Stats for ou=people,dc=example,dc=com ---

Index : Total Bytes : Key Bytes : Keys : Size (# Keys) : Inserted :
Removed : Replaced: Hits : Misses : Discarded : Duplicates
------:-------------:-----------:---------:---------------:----------
rdn   : 30667304    : 14888906  : 1000001 : 3464494 0 :0 :0 :0 :0 :0
uid   : 26523480    : 10888902  : 1000001 : 3464494 0 : 0 : 3583 : 0 : 0 : 0


          --- Operation Processing Time ---

Op Type   : Total Ops : Avg Resp Time (ms)
----------:-----------:-------------------
Add       : 0         : 0.0
Bind      : 0         : 0.0
Compare   : 0         : 0.0
Delete    : 0         : 0.0
Modify    : 0         : 0.0
Modify DN : 0         : 0.0
Search    : 3583      : 117.58
All       : 3583      : 117.58


          --- Work Queue ---

           : Recent : Average : Maximum
-----------:--------:---------:--------
Queue Size : 0      : 0       : 1
% Busy     : 0      : 1       : 19
```

About the Monitor Entries

While the Directory Proxy Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the `cn=monitor` subtree. The types of monitor entries that are available include:

- **General Monitor Entry (cn=monitor)** – Provides a basic set of general information about the server.
- **Active Operations Monitor Entry (cn=Active Operations,cn=monitor)** – Provides information about all operations currently in progress in the server.
- **Backend Monitor Entries (cn={id} Backend,cn=monitor)** – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- **Client Connections Monitor Entry (cn=Client Connections,cn=monitor)** – Provides information about all connections currently established to the server.
- **Connection Handler Monitor Entry (cn={name},cn=monitor)** – Provides information about the configuration of each connection handler and the client connections established to it.
- **Database Environment Monitor Entries (cn={id} Database Environment,cn=monitor)** – Provides statistics and other data from the Oracle Berkeley DB Java Edition database envi- ronment used by the associated backend.
- **Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor)** – Provides infor- mation about the amount of usable disk space available to server components.
- **JVM Memory Usage Monitor Entry (cn=JVM Memory Usage,cn=monitor)** – Provides information about garbage collection activity, the amount of memory available to the server, and the amount of memory consumed by various server components.
- **JVM Stack Trace Monitor Entry (cn=JVM Stack Trace,cn=monitor)** – Provides a stack trace of all threads in the JVM.
- **LDAP Statistics Monitor Entries (cn={name} Statistics,cn=monitor)** – Provides information about the number of each type of operation requested and bytes transferred over the connection handler.
- **Processing Time Histogram Monitor Entry (cn=Processing Time Histogram,cn=monitor)** – Provides information about the number of percent of operations that completed in various response time categories.
- **SSL Context Monitor Entry (cn=SSL Context,cn=monitor)** – Provides information about the available and supported SSL Cipher Suites and Protocols on the server.
- **System Information Monitor Entry (cn=System Information,cn=monitor)** – Provides information about the underlying JVM and system.
- **Version Monitor Entry (cn=Version,cn=monitor)** – Provides information about the Directory Proxy Server version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Directory Proxy Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Using the Monitoring Interfaces

The PingDirectoryProxy Server exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the Administrative Console, JMX, over LDAP, using the LDAP SDK, and using SNMP.

Monitoring with the Administrative Console

Ping Identity has an Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Proxy Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.
4. Use the top level navigation dropdown and select 'Status.'

5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

The PingDirectoryProxy Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the Administrative Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The PingDirectoryProxy Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Proxy Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Proxy Server using the `dsconfig` tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

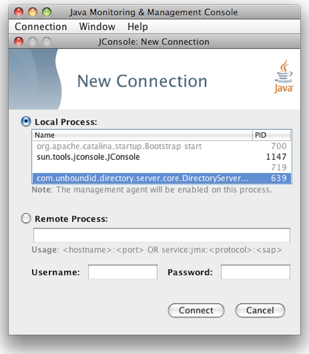
To Run JConsole

1. Use `JConsole` to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: `jconsole PID`. Or you can run JConsole remotely using: `jconsole hostname:port`.

   ```
   $ jconsole
   ```

   Note: If SSL is configured on the JMX Connection Handler, you must specify the Directory Proxy Server jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

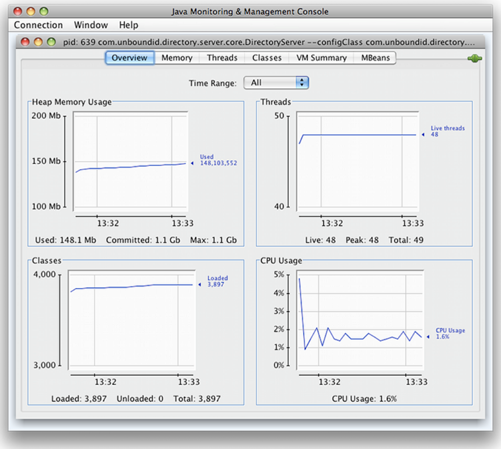
   ```
   $ jconsole \
     -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
     -J-Djavax.net.ssl.trustStorePassword=secret \
     -J-Djava.class.path=$SERVER_ROOT/lib/PingDirectoryProxy.jar:/
   Library/Java/JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/
   jconsole.jar
   ```
2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.



Name	PID
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1147
	719
com.unboundid.directory.server.core.DirectoryServer...	639

Note: The management agent will be enabled on this process.

3. Review the resource monitoring information.



Used: 148.1 Mb   Committed: 1.1 Gb   Max: 1.1 Gb

Live: 48   Peak: 48   Total: 49

Loaded: 3,897   Unloaded: 0   Total: 3,897

CPU Usage: 1.6%

Monitoring the Directory Proxy Server Using JConsole

You can set up JConsole to monitor the Directory Proxy Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscibe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Proxy Server using JConsole

1. Start the Directory Proxy Server.

```
$ bin/start-server
```

2. Enable the JMX Connection handler using the `dsconfig` tool. The handler is disabled by default. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

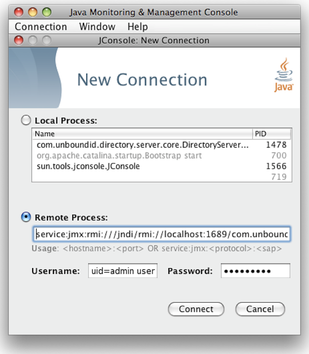
```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
```

```
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Proxy Server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



○ Local Process:

Name	PID
com.unboundid.directory.server.core.DirectoryServer...	1478
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1566
	719

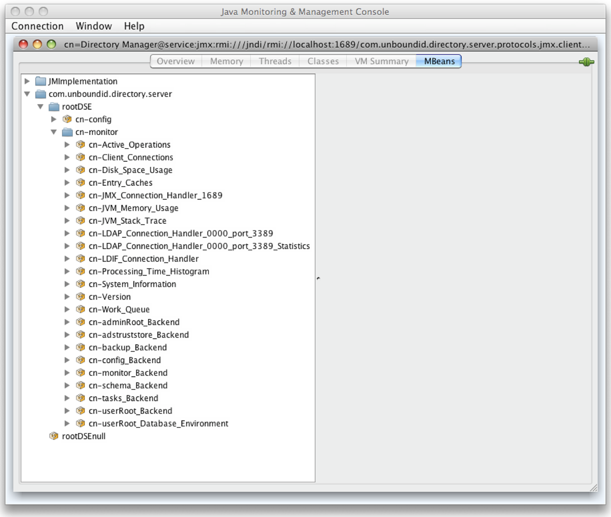
● Remote Process:

service:jmx:rmi:///jndi/rmi://localhost:1689/com.unbounc

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

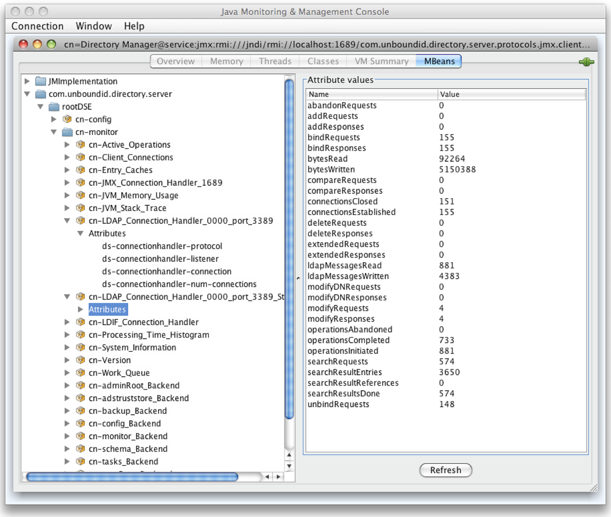
Username: uid=admin user Password: ••••••••••

6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.


Java Monitoring & Management Console
Connection
Window
Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview
Memory
Threads
Classes
VM Summary
MBeans

```
▶ JMImplementation
▼ com.unboundid.directory.server
    ▼ rootDSE
        ▶ cn-config
        ▼ cn-monitor
            ▶ cn-Active_Operations
            ▶ cn-Client_Connections
            ▶ cn-Disk_Space_Usage
            ▶ cn-Entry_Caches
            ▶ cn-JMX_Connection_Handler_1689
            ▶ cn-JVM_Memory_Usage
            ▶ cn-JVM_Stack_Trace
            ▶ cn-LDAP_Connection_Handler_0000_port_3389
            ▶ cn-LDAP_Connection_Handler_0000_port_3389_Statistics
            ▶ cn-LDIF_Connection_Handler
            ▶ cn-Processing_Time_Histogram
            ▶ cn-System_Information
            ▶ cn-Version
            ▶ cn-Work_Queue
            ▶ cn-adminRoot_Backend
            ▶ cn-adstruststore_Backend
            ▶ cn-backup_Backend
            ▶ cn-config_Backend
            ▶ cn-monitor_Backend
            ▶ cn-schema_Backend
            ▶ cn-tasks_Backend
            ▶ cn-userRoot_Backend
            ▶ cn-userRoot_Database_Environment
    rootDSEnull
```

7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.



cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...

- JMImplementation
- com.unboundid.directory.server
  - rootDSE
    - cn-config
    - cn-monitor
      - cn-Active_Operations
      - cn-Client_Connections
      - cn-Entry_Caches
      - cn-JMX_Connection_Handler_1689
      - cn-JVM_Memory_Usage
      - cn-JVM_Stack_Trace
      - cn-LDAP_Connection_Handler_0000_port_3389
        - Attributes
          - ds-connectionhandler-protocol
          - ds-connectionhandler-listener
          - ds-connectionhandler-connection
          - ds-connectionhandler-num-connections
      - cn-LDAP_Connection_Handler_0000_port_3389_St
        - Attributes
      - cn-LDIF_Connection_Handler
      - cn-Processing_Time_Histogram
      - cn-System_Information
      - cn-Version
      - cn-Work_Queue
      - cn-adminRoot_Backend
      - cn-adstruststore_Backend
      - cn-backup_Backend
      - cn-config_Backend
      - cn-monitor_Backend
      - cn-schema_Backend
      - cn-tasks_Backend

Attribute values

Name	Value
abandonRequests	0
addRequests	0
addResponses	0
bindRequests	155
bindResponses	155
bytesRead	92264
bytesWritten	5150388
compareRequests	0
compareResponses	0
connectionsClosed	151
connectionsEstablished	155
deleteRequests	0
deleteResponses	0
extendedRequests	0
extendedResponses	0
ldapMessagesRead	881
ldapMessagesWritten	4383
modifyDNRequests	0
modifyDNResponses	0
modifyRequests	4
modifyResponses	4
operationsAbandoned	0
operationsCompleted	733
operationsInitiated	881
searchRequests	574
searchResultEntries	3650
searchResultReferences	0
searchResultsDone	574
unbindRequests	148

Monitoring over LDAP

The PingDirectoryProxy Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Proxy Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " +
 String.valueOf(value));
      }
     }
     System.out.println();
  }
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring Using SNMP

The PingDirectoryProxy Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Proxy Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Proxy Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems. Consult with your NMS system for specific information.

The PingDirectoryProxy Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Proxy Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Proxy Server’s SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

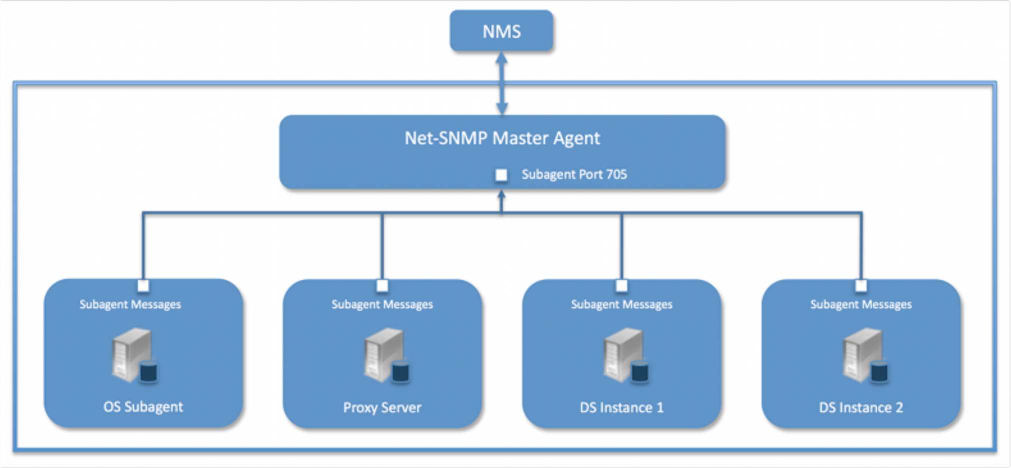


Figure 8: Example SNMP Deployment

One important note is that the PingDirectoryProxy Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note: The Directory Proxy Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Proxy Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Proxy Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" --set enabled:true
```

2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master
agent at localhost/705: Timeout
```

3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

```
master agentx agentXSocket tcp:localhost:705
```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

```
trap2sink localhost public 162
```

5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

```
rwuser initial
createUser initial MD5 setup_passphrase DES
```

6. Run the following command to create the SNMPv3 user.

```
snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
localhost create snmpuser initial
```

7. Start the `snmpd` daemon and after a few seconds you should see the following message in the Directory Proxy Server error log:

```
The SNMP subagent connected successfully to the master agent
at localhost:705. The SNMP context name is host.example.com:389
```

8. Set up a trap client to see the alerts that are generated by the Directory Proxy Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

```
authcommunity log, execute public
```

9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

10. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (-f), do not read any configuration files (-C) except the one specified with the -c option, print to standard output (-Lo), and then specify that debugging output

should be turned on for the User-based Security Module (-Dusm). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
  -m all -M +/usr/share/snmp/mibs
```

11. Run the Net-SNMP client tools to test the feature. The following options are required: -v <SNMP version>, -u <user name>, -A <user password>, -l <security level>, -n <context name (instance name)> . The -m all option loads all MIBs in the default MIB directory in /usr/share/snmp/mibs so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389
 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n
 host.example.com:389 \
-m all localhost systemStatus
```

MIBS

The Directory Proxy Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Proxy Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Proxy Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Profiling Server Performance Using the Stats Logger

The Directory Proxy Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Directory Proxy Server ships with the built-in "Stats Logger' disabled. To enable it using the `dsconfig` tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select .

1. Run `dsconfig` in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

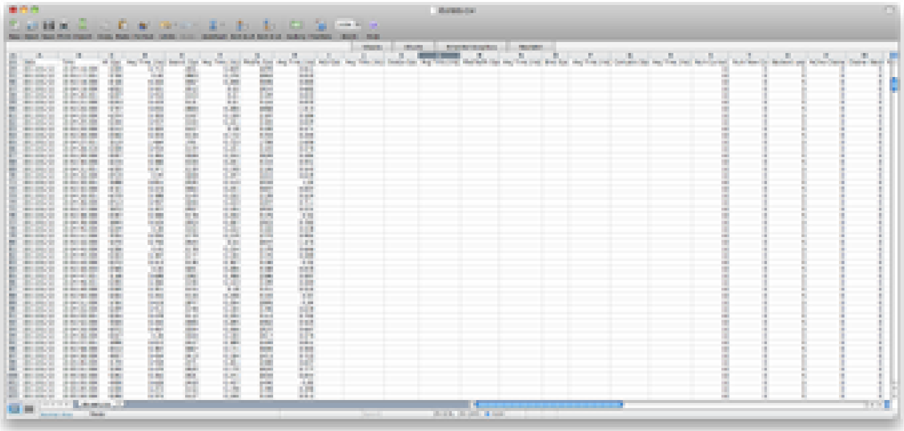
```
$ bin/dsconfig
```

2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plug-in.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

```
>>>> Configure the properties of the Stats Logger Plugin

Property                Value(s)
------------------------------------------------------------------------
1)   description        Logs performance stats to a log file
                        periodically.
2)   enabled                     false
3)   local-db-backend-info       basic
4)   replication-info            basic
5)   entry-cache-info            -
6)   host-info                   -
7)   included-ldap-application   If per-application LDAP stats is enabled,
                                 then stats will be included for all
                                 applications.
8)   log-interval                1 s
9)   collection-interval         200 ms
10)  suppress-if-idle            true
11)  header-prefix-per-column    false
12)  empty-instead-of-zero       true
13)  lines-between-header        50
14)  included-ldap-stat          active-operations, num-connections,
                                 op-count-and-latency, work-queue
15)  included-resource-stat      memory-utilization
16)  histogram-format            count
17)  histogram-op-type           all
18)  per-application-ldap-stats  aggregate-only
19)  ldap-changelog-info         -
20)  gauge-info                  none
21)  log-file                    logs/dsstats.csv
22)  log-file-permissions        640
23)  append                      true
24)  rotation-policy             Fixed Time Rotation Policy, Size Limit
                                 Rotation Policy
25)  retention-policy            File Count Retention Policy

?)   help
f)   finish - apply any changes to the Periodic Stats Logger Plugin
a)   hide advanced properties of the Periodic Stats Logger Plugin
d)   display the equivalent dsconfig command lines to either re-create this
```

```
              object or only to apply pending changes
b)    back
q)    quit

Enter choice [b]:
```

7. Run the Directory Proxy Server. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate` --help to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in *To Enable the Stats Logger*.
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. On the Directory Proxy Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

    ```
    >>>> Configure the properties of the Custom Logged Stats
     >>>> via creating 'Memory Usage' Custom Logged Stats

             Property                    Value(s)
             ---------------------------------------------------------------
        1)   description                 -
    ```

```
    2)   enabled                     true
    3)   monitor-objectclass         ds-memory-usage-monitor-entry
    4)   include-filter              -
    5)   attribute-to-log            total-bytes-used-by-memory-consumers
    6)   column-name                 Memory Consumer Total (GB)
    7)   statistic-type              raw
    8)   header-prefix               -
    9)   header-prefix-attribute     -
    10)  regex-pattern               -
    11)  regex-replacement           -
    12)  divide-value-by             1073741824
    13)  divide-value-by-attribute  -
    14)  decimal-format              #.##
    15)  non-zero-implies-not-idle  false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

Working with Alarms, Alerts, and Gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server unavailable' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and Administrative Console. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The Directory Proxy Server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

The Directory Proxy Server is compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when the Condition and Resource properties are the same. The Condition corresponds to the Summary column in the `admin-alerts-list.csv` file.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms.

To Test Alarms and Alerts

1. Configure a gauge with `dsconfig` and set the `override-severity` property to critical. The following example uses the CPU Usage (Percent) gauge.

```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

2. Run the `status` tool to verify that an alarm was generated with corresponding alerts. The `status` tool provides a summary of the server’s current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status

                        --- Administrative Alerts ---
 Severity : Time            : Message
 ---------:-----------------:-------------------------------------------------------
 Info     : 11/Aug/2014     : A configuration change has been made in the
          : 15:48:46 -0500  : Directory Server:
          :                 : [11/Aug/2014:15:48:46.054 -0500]
          :                 : conn=17 op=73 dn='cn=Directory Manager,cn=Root
          :                 : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                 : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                 :  --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                 : --set warning-value:-1'
 Info     : 11/Aug/2014     : A configuration change has been made in the
          :  15:47:32 -0500 : Directory Server: [11/Aug/2014:15:47:32.547
 -0500]
          :                 : conn=4 op=196 dn='cn=Directory Manager,cn=Root
          :                 : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                 : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                 : --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                 :  --set warning-value:0'
 Error    : 11/Aug/2014     : Alarm [CPU Usage (Percent). Gauge CPU Usage
 (Percent)
          :  15:41:00 -0500 : for Host System has
          :                 : a current value of '18.583333333333332'.
          :                 : The severity is currently OVERRIDDEN in the
          :                 : Gauge's configuration to 'CRITICAL'.
          :                 : The actual severity is: The severity is
          :                 : currently 'NORMAL', having assumed this
 severity
          :                 : Mon Aug 11 15:41:00 CDT 2014. If CPU use is
 high,
          :                 : check the server's current workload and make
 any
          :                 : needed adjustments. Reducing the load on the
 system
          :                 : will lead to better response times.
          :                 : Resource='Host System']
          :                 : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
 hours
 Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
                         --- Alarms ---
 Severity : Severity Start : Condition : Resource    : Details
          : Time           :           :             :
 ---------:----------------:-----------:-------------:------------------------------
 Critical : 11/Aug/2014    : CPU Usage : Host System : Gauge CPU Usage
 (Percent) for
          : 15:41:00 -0500 : (Percent) :             : Host System
          :                :           :             : has a current value
 of
          :                :           :             : '18.785714285714285'.
          :                :           :             : The severity is
 currently
          :                :           :             : 'CRITICAL', having
 assumed
          :                :           :             : this severity Mon Aug
 11
          :                :           :             : 15:49:00 CDT 2014. If
 CPU use
          :                :           :             : is high, check the
 server's
          :                :           :             : current workload and
 make any
          :                :           :             : needed adjustments.
 Reducing
          :                :           :             : the load on the
 system will
          :                :           :             : lead to better
 response times
 Warning  : 11/Aug/2014    : Work Queue: Work Queue  : Gauge Work Queue Size
 (Number
          : 15:39:40 -0500 : Size      :             : of Requests) for Work
 Queue
          :                : (Number of:             : has a current value
 of '27'.
          :                : Requests) :             : The severity is
 currently
          :                :           :             : 'WARNING' having
 assumed this
          :                :           :             : severity Mon Aug 11
 15:48:50
```

```
          :                :           :             : CDT 2014. If all
 worker
          :                :           :             : threads are busy
 processing
          :                :           :             : other client
 requests, then
          :                :           :             : new requests that
 arrive will
          :                :           :             : be forced to wait in
 the work
          :                :           :             : queue until a worker
 thread
          :                :           :             : becomes available
Shown are alarms of severity [Warning,Minor,Major,Critical]
Use the --alarmSeverity option to filter this list
```

Indeterminate Alarms

Indeterminate alarms are raised for a server condition for which a severity cannot be determined. In most cases these alarms are benign and do not issue alerts nor appear in the output of the `status` tool or Administrative Console by default. These alarms are usually caused by an enabled gauge that is intended to measure an aspect of the server that is not currently enabled. For example, gauges intended to monitor metrics related to replication may produce indeterminate alarms if a Directory Server is not currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There may be messages that can help determine the issue. The following is sample output from the `status` tool run with the –`alarmSeverity=indeterminate` option:

```
                        --- Alarms ---
Severity     : Severity Start : Condition      : Resource   : Details
             : Time           :                :            :
-------------:----------------:----------------:------------:------------------------
Normal       : 26/Aug/2014    : Startup Begun  : cn=config  : The Directory
 Server
             : 14:16:29 -0500 :                :            : is starting.
             :                :                :            :
Indeterminate: 26/Aug/2014    : Replication    : not        : The value of
 gauge
             : 14:16:40 -0500 : Latency        : available  : Replication
 Latency
             :                : (Milliseconds) :            : (Milliseconds)
 could not
             :                :                :            : be determined.
 The
             :                :                :            : severity is
 INDETERMINATE,
             :                :                :            : having assumed
 this
             :                :                :            : severity Tue Aug
 26
             :                :                :            : 14:17:10 CDT
 2014.
```

The following is an indeterminate alarm for the Replication Latency (Milliseconds) gauge. The following is a sample search of the monitor backend for this gauge’s entry. The result is an error message may explain the indeterminate severity:

```
# ldapsearch -w password --baseDN "cn=monitor"  \
-D"cn=directory manager" gauge-name="Replication Latency (Milliseconds)"

dn: cn=Gauge Replication Latency (Milliseconds),cn=monitor
objectClass: top
```

```
objectClass: ds-monitor-entry
objectClass: ds-numeric-gauge-monitor-entry
objectClass: ds-gauge-monitor-entry
objectClass: extensibleObject
cn:          Gauge Replication Latency (Milliseconds)
gauge-name:  Replication Latency (Milliseconds)
resource:
severity:    indeterminate
summary:     The value of gauge Replication Latency (Milliseconds) could not
             be determined. The severity is INDETERMINATE, having assumed
             this severity Tue Aug 26 15:42:40 CDT 2014
error-message: No entries were found under cn=monitor having object
               class ds-replica-monitor-entry
             …
```

Working with Administrative Alert Handlers

The PingDirectoryProxy Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Proxy Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler**. Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Directory Proxy Server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler**. Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. Ping Identity uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler**. Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler**. Provides alert handler implementations created in third-party code using the Server SDK.

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the `dsconfig` tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using `dsconfig` in interactive command-line mode, which is visible on the "Standard" object menu.

To Configure the JMX Connection Handler

1. Use `dsconfig` to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the `ldapmodify` tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
givenName: JMX
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

To Configure the JMX Alert Handler

- Use `dsconfig` to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the `dsconfig` tool.

Configuring the SMTP Alert Handler

- Use the `dsconfig` tool to configure the SMTP Alert Handler.

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the `dsconfig` tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP2c. For more information on SNMP, see Monitoring Using SNMP.

To Configure the SNMP Subagent Alert Handler

- Use `dsconfig` to configure the SNMP subagent alert handler. The `server-host-name` is the address of the system running the SNMP subagent. The `server-port` is the port number on which the subagent is running. The `community-name` is the name of the SNMP community that is used for the traps.

  The Directory Proxy Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

Working with Virtual Attributes

The PingDirectoryProxy Server provides dynamically generated attributes called virtual attributes for local Directory Proxy Server data. The proxy virtual attributes apply to a local proxy backend, such as `cn=config` or the Root DSE. If you want to have virtual attributes in entries for proxied requests, then they must be configured in the backend servers. Alternately, attributes may be inserted into those entries using proxy transformations. For more information about configuring proxy transformations, see “Configuring Proxy Transformations”.

For example, you can define a virtual attribute and assign it to the Root DSE as follows:

```
$ bin/dsconfig create-virtual-attribute \
  --name defineDescriptionOnRootDSE --type user-defined \
  --set enabled:true --set attribute-type:description \
  --set filter:objectclass=ds-root-dse --set value:PrimaryProxy
```

If you search the Root DSE using the following LDAP search, you see that the description attribute now has the value `PrimaryProxy`.

```
$ bin/ldapsearch --baseDN "" --searchScope base --bindDN "" \
  --bindPassword "" --port 5389 -- hostname localhost \
  "objectclass=*" description

dn:
description:PrimaryProxy
```

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Proxy Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Chapter 9

Managing Monitoring

Topics:

The PingDirectoryProxy Server also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces via the PingDataMetrics™ Server, the Administrative Console, SNMP, JMX, and over LDAP. The Directory Proxy Server also provides a tool, the Periodic Stats Logger, to profile server performance.

This chapter presents the following information:

The Monitor Backend

The Directory Proxy Server exposes its monitoring information under the `cn=monitor` entry. Administrators can use various means to monitor the servers, including the PingData Metrics Server, through SNMP, the Administrative Console, JConsole, LDAP command-line tools, and the Periodic Stats Logger. Use the `bin/status` tool to display server component activity and state.

The list of all monitor entries can be seen using ldapsearch as follows:

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
 --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
 --baseDN "cn=monitor" "(objectclass=*)" cn
```

The following table describes a subset of the monitor entries:

Table 11: Directory Proxy Server Monitoring Components

Component	Description
Active Operations	Provides information about the operations currently being processed by the Directory Proxy Server. Shows the number of operations, information on each operation, and the number of active persistent searches.
Backends	Provides general information about the state of an a Directory Proxy Server backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size.
Client Connections	Provides information about all client connections to the Directory Proxy Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., connID="15", connectTime="20100308223038Z", etc.)
Connection Handlers	Provides information about the available connection handlers on the Directory Proxy Server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
Disk Space Usage	Provides information about the disk space available to various components of the Directory Proxy Server.
General	Provides general information about the state of the Directory Proxy Server, including product name, vendor name, server version, etc.
Index	Provides on each index. The monitor captures the number of keys preloaded, and counters for read/write/remove/open-cursor/read-for-search. These counters provide insight into how useful an index is for a given workload.
HTTP/HTTPS Connection Handler Statistics	Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc.

Component	Description
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc.
System Information	Provides general information about the system and the JVM on which the Directory Proxy Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Directory Proxy Server version, including build ID, version, revision number, etc.
Work Queue	Provides information about the state of the Directory Proxy Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, and number of busy worker threads. The work queue configuration has a `monitor-queue-time` property set to `true` by default. This logs messages for new operations with a `qtime` attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

Monitoring Disk Space Usage

The disk space usage monitor provides information about the amount of usable disk space available for Directory Proxy Server components. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface by various components of the server. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `/config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the DiskSpaceConsumer interface.

The disk space usage monitor provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

Three thresholds can be configured for this monitor:

- **Low space warning threshold**. This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Directory Proxy Server will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The Directory Proxy Server will also generate additional alerts as the amount of usable space is further reduced (e.g., each time the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.
- **Low space error threshold**. This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. Once the server enters this mode, then an administrator will have to

take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server will resume normal operation. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.

- **Out of space error threshold**. This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the PingDirectoryProxy Server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.
- **Disk space monitoring for tools**. The server monitors disk space consumption during processing for the `export-ldif`, `rebuild-index`, and `backup` tools. Space is monitored every 10 seconds if usable space for all monitored paths is greater than 15 percent of the capacity of those volumes. If usable space for any path drops below 15 percent, or below 10GB free, the space check frequency is increased to every second. Warning messages are generated if available space falls below 10 percent, or below 5GB free. If usable space for any path drops below two percent, or 1GB free, the tool processing is aborted and files may be removed to free up space.

The default configuration uses the same values for the low space error threshold and out of space error threshold. This is to prevent having the server online but rejecting requests, which will cause problems with applications trying to interact with the server. The low space warning threshold generates an alert before the problem becomes serious, well in advance of available disk space dropping to a point that it is critical.

The default values may not be suitable for all disk sizes, and should be adjusted to fit the deployment. Determining the best values should factor in the size of the disk, how big the database may become, how much space log files may consume, and how many backups will be stored.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring with the PingDataMetrics Server

The PingDataMetrics Server is an invaluable tool for collecting, aggregating and exposing historical and instantaneous data from the various Ping Identity servers in a deployment. The PingDataMetrics Server relies on a captive PostgreSQL data store for the metrics, which it collects from internal instrumentation across the instances, replicas, and data centers in your environment. The data is available via a Monitoring API that can be used to build custom dashboards and monitoring applications to monitor the overall health of your Ping Identity Platform system. For more information, see the *PingDataMetrics Server Administration Guide*.

Monitoring Key Performance Indicators by Application

The PingDirectoryProxy Server can be configured to track many key performance metrics (for example, throughput and response-time) by the client applications requesting them. This feature is invaluable for measuring whether the Ping Identity identify infrastructure meets all of your service-level agreements (SLA) that have been defined for client applications.

When enabled, the per-application monitoring data can be accessed in the `cn=monitor` backend, the Periodic Stats Logger, and made available for collection by the Metrics Server. See the “Profiling Server Performance Using the Periodic Stats Logger” for more information on using that component. Also, see the Directory Proxy Server Configuration section of the *PingData Metrics Server Administration Guide* for details on configuring the server to expose metrics that interest you. Tracked application information is exposed in the PingDataMetrics Server by metrics having the 'application-name' dimension. See the documentation under `docs/metrics` of the PingDataMetrics Server for information on which metrics are available with the 'application-name' dimension.

Configuring the External Servers

Before you install the PingDataMetrics Server, you need to configure the servers you will be monitoring: PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. The PingDataMetrics Server requires all servers to be version 3.5.0 or later. See the administration guides for each product for installation instructions.

Once you have installed the Directory Proxy Server, you can use the `dsconfig` tool to make configuration changes for the PingDataMetrics Server. When using the `dsconfig` tool interactively, set the complexity level to Advanced, so that you can make all the necessary configuration changes.

Preparing the Servers Monitored by the PingDataMetrics Server

The Metrics Backend manages the storage of metrics and provides access to the stored blocks of metrics via LDAP. The Metrics Backend is configured to keep a maximum amount of metric history based on log retention policies. The default retention policy uses the Default Size Limit Retention Policy, Free Disk Space Retention Policy, and the File Growth Limit Policy, limiting the total disk space used to 500 MB. This amount of disk typically contains more than 24 hours of metric history, which is ample. The Directory Proxy Server keeps a metric history so that the PingDataMetrics Server can be down for a period and then catch up when it comes back online.

The following two commands create a Retention Policy that limits the number of files to 2000, and sets the Metrics Backend to flush data to a new file every 30 seconds.

```
$ bin/dsconfig create-log-retention-policy \
  --policy-name StatsCollectorRetentionPolicy \
  --type file-count --set number-of-files:2000

$ bin/dsconfig set-backend-prop \
  --backend-name metrics --set sample-flush-interval:30s \
  --set retention-policy:StatsCollectorRetentionPolicy
```

These commands configure the Metrics Backend to keep 16 hours of metric history, which consumes about 250 MB of disk, ensuring that captured metrics are available to the PingDataMetrics Server within 30 seconds of when the metric was captured. The value of the `sample-flush-interval` attribute determines the maximum delay between when a metric is captured and when it can be picked up by the PingDataMetrics Server.

The flush interval can be set between 15 seconds and 60 seconds, with longer values resulting in less processing load on the PingDataMetrics Server. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in the Dashboard Application. If you change the `sample-flush-interval` attribute to 60 seconds in the example above, then the Directory Proxy Server keeps 2000 minutes of history. Because the number of metrics produced per unit of time can vary depending on the configuration, no exact formula can be used to compute how much storage is required for each hour of history. However, 20 MB per hour is a good estimate.

Configuring the Processing Time Histogram Plugin

The Processing Time Histogram plugin is configured on each Directory Proxy Server and Directory Proxy Server as a set of histogram bucket ranges. When the bucket ranges for a histogram change, the PingDataMetrics Server notices the change and marks samples differently. This process allows for histograms with the same set of bucket definitions to be properly aggregated and understood when returned in a query. If different servers have different bucket definitions, then a single metric query cannot return histogram data from the servers.

You should try to keep the Processing Time Histogram bucket definitions the same on all servers. Having different definitions restricts the ability of the PingDataMetrics Server API to aggregate histogram data across servers and makes the results of a query asking "What percentage of the search requests took less than 12 milliseconds?" harder to understand.

For each server in your topology, you must set the separate-monitor-entry-per-tracked-application property of the processing time histogram plugin to true. This property must be set to expose per-application monitoring information under `cn=monitor`. When the separate-monitor-entry-per-tracked-application property is set to true, then the per-application-ldap-stats property must be set to per-application-only on the Stats Collector Plugin and vice versa.

For example, the following `dsconfig` command line sets the required properties of the Processing Time Histogram plugin:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Processing Time Histogram” \
  --set separate-monitor-entry-per-tracked-application:true
```

The following `dsconfig` command line sets the per-application-ldap-stats property of the Stats Collector plugin to per-application-only:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Stats Collector” \
 --set per-application-ldap-stats:per-application-only
```

Setting the Connection Criteria to Collect SLA Statistics by Application

If you want to collect data about your SLAs, you need to configure connection criteria for each Service Level Agreement that you want to track. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used when the server needs to perform matching based on connection-level properties, such as filtered logging. For assistance using connection criteria, contact your authorized support provider.

For example, imagine that we are interested in collecting statistics on data that is accessed by clients authenticating as the Directory Manager. We need to create connection criteria on the Directory Proxy Server that identifies any user authenticating as the Directory Manager. The connection criteria name corresponds to the application-name dimension value that clients will specify when accessing the data via the API. When you define the Connection Criteria, change the included-user-base-dn property to include the Directory Manager’s full LDIF entry.

The following `dsconfig` command line creates connection criteria for the Directory Manager:

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name “Directory Manager” \
  --type simple \
  --set “included-user-base-dn:cn=Directory Manager,cn=Root DNs,cn=config”
```

Updating the Global Configuration

You also need to create Global Configuration-tracked applications for each app (connection criteria) you intend to track. The `tracked-application` property allows individual applications to be identified in the server by connection criteria. The name of the tracked application is the same as the name you defined for the connection criteria.

For example, the following `dsconfig` command line adds the connection criteria we created in the previous step to the list of tracked applications:

```
$ bin/dsconfig set-global-configuration-prop \
  --set "tracked-application:Directory Manager”
```

The value of the `tracked-application` field corresponds to the value of the `application-name` dimension value that clients will specify when accessing the data via the API.

Proxy Considerations for Tracked Applications

In a proxy environment, the criteria should be defined in the Directory Proxy Server since the Directory Proxy Server passes the application name through to the Directory Server in the intermediate client control. If a client of the Directory Proxy Server or Directory Server happens to use the intermediate client control, then the client name specified in the control will be used as the application name regardless of the criteria listed in the tracked-application property.

Monitoring Using SNMP

The PingDirectoryProxy Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Proxy Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Proxy Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems. Consult with your NMS system for specific information.

The PingDirectoryProxy Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Proxy Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Proxy Server's SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

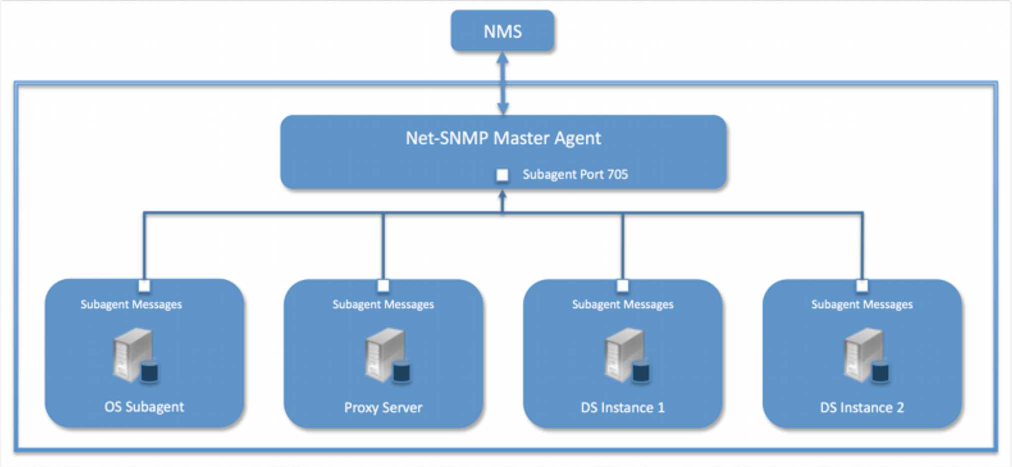


Figure 9: Example SNMP Deployment

One important note is that the PingDirectoryProxy Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default.

Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note: The Directory Proxy Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Proxy Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Proxy Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" --set enabled:true
```

2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master
agent at localhost/705: Timeout
```

3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

```
master agentx agentXSocket tcp:localhost:705
```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

```
trap2sink localhost public 162
```

5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

```
rwuser initial
createUser initial MD5 setup_passphrase DES
```

6. Run the following command to create the SNMPv3 user.

```
snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
localhost create snmpuser initial
```

7. Start the `snmpd` daemon and after a few seconds you should see the following message in the Directory Proxy Server error log:

```
The SNMP subagent connected successfully to the master agent
at localhost:705. The SNMP context name is host.example.com:389
```

8. Set up a trap client to see the alerts that are generated by the Directory Proxy Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

```
authcommunity log, execute public
```

9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

10. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (-f), do not read any configuration files (-C) except the one specified with the -c option, print to standard output (-Lo), and then specify that debugging output should be turned on for the User-based Security Module (-Dusm). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

    ```
    $ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
      -m all -M +/usr/share/snmp/mibs
    ```

11. Run the Net-SNMP client tools to test the feature. The following options are required: -v <SNMP version>, -u <user name>, -A <user password>, -l <security level>, -n <context name (instance name)> . The -m all option loads all MIBs in the default MIB directory in /usr/share/snmp/mibs so that MIB names can be used in place of numeric OIDs.

    ```
    $ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389
     \
    -m all localhost localDBBackendCount.0

    $ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n
     host.example.com:389 \
    -m all localhost systemStatus
    ```

MIBS

The Directory Proxy Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Proxy Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Proxy Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Monitoring with the Administrative Console

Ping Identity has an Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Proxy Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.
4. Use the top level navigation dropdown and select 'Status.'
5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

The PingDirectoryProxy Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the Administrative Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The PingDirectoryProxy Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Proxy Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Proxy Server using the `dsconfig` tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

To Run JConsole

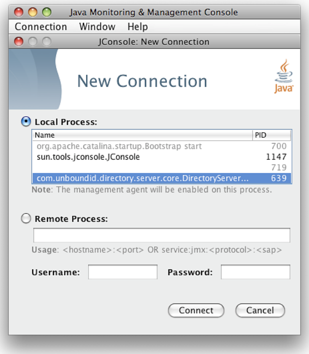
1. Use `JConsole` to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: `jconsole PID`. Or you can run JConsole remotely using: `jconsole hostname:port`.

```
$ jconsole
```

Note: If SSL is configured on the JMX Connection Handler, you must specify the Directory Proxy Server jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

```
$ jconsole \
  -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
  -J-Djavax.net.ssl.trustStorePassword=secret \
  -J-Djava.class.path=$SERVER_ROOT/lib/PingDirectoryProxy.jar:/
Library/Java/JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/
jconsole.jar
```

2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.

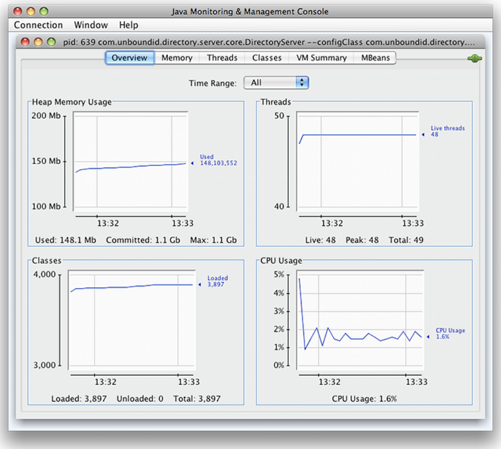


Name	PID
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1147
	719
com.unboundid.directory.server.core.DirectoryServer...	639

Note: The management agent will be enabled on this process.

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

3. Review the resource monitoring information.



pid: 639 com.unboundid.directory.server.core.DirectoryServer --configClass com.unboundid.directory....

Time Range: All

Used: 148.1 Mb Committed: 1.1 Gb Max: 1.1 Gb

Live: 48 Peak: 48 Total: 49

Loaded: 3,897 Unloaded: 0 Total: 3,897

CPU Usage: 1.6%

Monitoring the Directory Proxy Server Using JConsole

You can set up JConsole to monitor the Directory Proxy Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscibe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Proxy Server using JConsole

1. Start the Directory Proxy Server.

```
$ bin/start-server
```

2. Enable the JMX Connection handler using the `dsconfig` tool. The handler is disabled by default. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

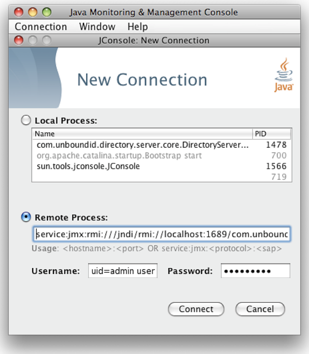
```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
```

```
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Proxy Server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



○ **Local Process:**

Name	PID
com.unboundid.directory.server.core.DirectoryServer...	1478
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1566
	719

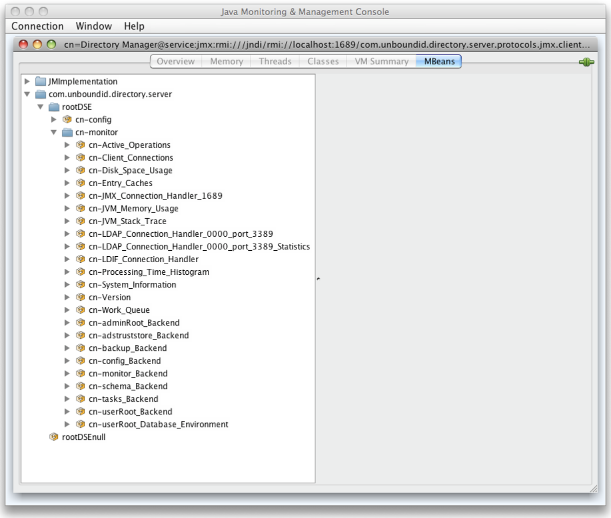
● **Remote Process:**

service:jmx:rmi:///jndi/rmi://localhost:1689/com.unbounc

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: uid=admin user **Password:** ••••••••••

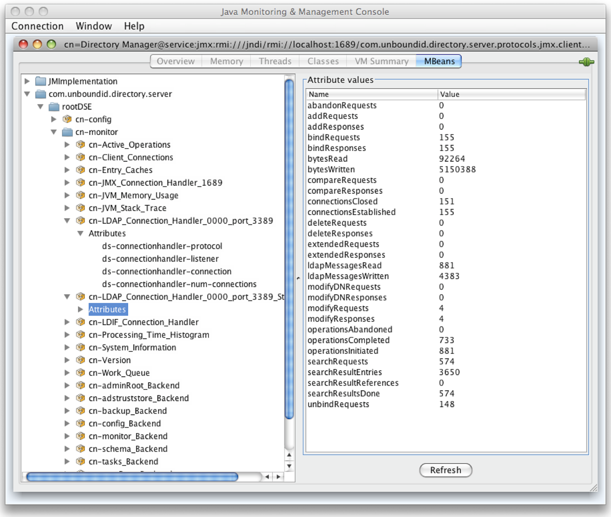
6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.



cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...

```
▶ JMImplementation
▼ com.unboundid.directory.server
  ▼ rootDSE
    ▶ cn-config
    ▼ cn-monitor
      ▶ cn-Active_Operations
      ▶ cn-Client_Connections
      ▶ cn-Disk_Space_Usage
      ▶ cn-Entry_Caches
      ▶ cn-JMX_Connection_Handler_1689
      ▶ cn-JVM_Memory_Usage
      ▶ cn-JVM_Stack_Trace
      ▶ cn-LDAP_Connection_Handler_0000_port_3389
      ▶ cn-LDAP_Connection_Handler_0000_port_3389_Statistics
      ▶ cn-LDIF_Connection_Handler
      ▶ cn-Processing_Time_Histogram
      ▶ cn-System_Information
      ▶ cn-Version
      ▶ cn-Work_Queue
      ▶ cn-adminRoot_Backend
      ▶ cn-adstruststore_Backend
      ▶ cn-backup_Backend
      ▶ cn-config_Backend
      ▶ cn-monitor_Backend
      ▶ cn-schema_Backend
      ▶ cn-tasks_Backend
      ▶ cn-userRoot_Backend
      ▶ cn-userRoot_Database_Environment
  rootDSEnull
```

7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.


Java Monitoring & Management Console
Connection
Window
Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview
Memory
Threads
Classes
VM Summary
MBeans
JMImplementation
com.unboundid.directory.server
rootDSE
cn-config
cn-monitor
cn-Active_Operations
cn-Client_Connections
cn-Entry_Caches
cn-JMX_Connection_Handler_1689
cn-JVM_Memory_Usage
cn-JVM_Stack_Trace
cn-LDAP_Connection_Handler_0000_port_3389
Attributes
ds-connectionhandler-protocol
ds-connectionhandler-listener
ds-connectionhandler-connection
ds-connectionhandler-num-connections
cn-LDAP_Connection_Handler_0000_port_3389_St
Attributes
cn-LDIF_Connection_Handler
cn-Processing_Time_Histogram
cn-System_Information
cn-Version
cn-Work_Queue
cn-adminRoot_Backend
cn-adstruststore_Backend
cn-backup_Backend
cn-config_Backend
cn-monitor_Backend
cn-schema_Backend
cn-tasks_Backend
Refresh

Attribute values

Name	Value
abandonRequests	0
addRequests	0
addResponses	0
bindRequests	155
bindResponses	155
bytesRead	92264
bytesWritten	5150388
compareRequests	0
compareResponses	0
connectionsClosed	151
connectionsEstablished	155
deleteRequests	0
deleteResponses	0
extendedRequests	0
extendedResponses	0
ldapMessagesRead	881
ldapMessagesWritten	4383
modifyDNRequests	0
modifyDNResponses	0
modifyRequests	4
modifyResponses	4
operationsAbandoned	0
operationsCompleted	733
operationsInitiated	881
searchRequests	574
searchResultEntries	3650
searchResultReferences	0
searchResultsDone	574
unbindRequests	148

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Proxy Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " +
 String.valueOf(value));
      }
     }
     System.out.println();
  }
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring over LDAP

The PingDirectoryProxy Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Profiling Server Performance Using the Stats Logger

The Directory Proxy Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Directory Proxy Server ships with the built-in "Stats Logger' disabled. To enable it using the `dsconfig` tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select .

1. Run `dsconfig` in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

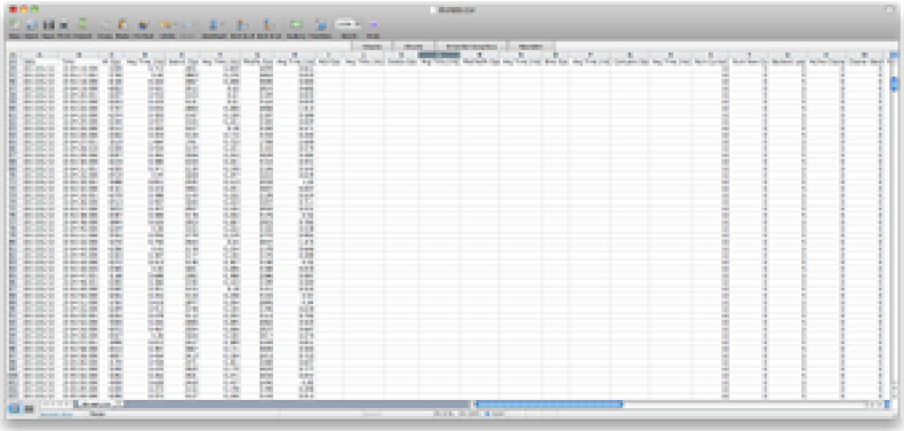
```
$ bin/dsconfig
```

2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plug-in.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

```
>>>> Configure the properties of the Stats Logger Plugin

Property                Value(s)
------------------------------------------------------------------------
1)   description        Logs performance stats to a log file
                        periodically.
2)   enabled                     false
3)   local-db-backend-info       basic
4)   replication-info            basic
5)   entry-cache-info            -
6)   host-info                   -
7)   included-ldap-application   If per-application LDAP stats is enabled,
                                 then stats will be included for all
                                 applications.
8)   log-interval                1 s
9)   collection-interval         200 ms
10)  suppress-if-idle            true
11)  header-prefix-per-column    false
12)  empty-instead-of-zero       true
13)  lines-between-header        50
14)  included-ldap-stat          active-operations, num-connections,
                                 op-count-and-latency, work-queue
15)  included-resource-stat      memory-utilization
16)  histogram-format            count
17)  histogram-op-type           all
18)  per-application-ldap-stats  aggregate-only
19)  ldap-changelog-info         -
20)  gauge-info                  none
```

```
21)   log-file                         logs/dsstats.csv
22)   log-file-permissions             640
23)   append                           true
24)   rotation-policy                  Fixed Time Rotation Policy, Size Limit
                                       Rotation Policy
25)   retention-policy                 File Count Retention Policy

?)    help
f)    finish - apply any changes to the Periodic Stats Logger Plugin
a)    hide advanced properties of the Periodic Stats Logger Plugin
d)    display the equivalent dsconfig command lines to either re-create this
            object or only to apply pending changes
b)    back
q)    quit

Enter choice [b]:
```

7. Run the Directory Proxy Server. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate` --help to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in *To Enable the Stats Logger*.
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. On the Directory Proxy Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.

13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                    Value(s)
         ---------------------------------------------------------------
    1)   description                 -
    2)   enabled                     true
    3)   monitor-objectclass         ds-memory-usage-monitor-entry
    4)   include-filter              -
    5)   attribute-to-log            total-bytes-used-by-memory-consumers
    6)   column-name                 Memory Consumer Total (GB)
    7)   statistic-type              raw
    8)   header-prefix               -
    9)   header-prefix-attribute     -
    10)  regex-pattern               -
    11)  regex-replacement           -
    12)  divide-value-by             1073741824
    13)  divide-value-by-attribute  -
    14)  decimal-format              #.##
    15)  non-zero-implies-not-idle  false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

Chapter 10

Troubleshooting the Directory Proxy Server

Topics:

This chapter provides the common problems and potential solutions that might occur when running PingDirectoryProxy Server. It is primarily targeted at cases in which the Directory Proxy Server is running on Linux® systems, but much of the information can be useful on other platforms as well.

This chapter presents the following information:

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the `jstat` utility with the -gc option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

To run the Directory Proxy Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`bin/start-server.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

Working with the Troubleshooting Tools

If problems arise with the Directory Proxy Server (whether from issues in the Directory Proxy Server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

Working with the Collect Support Data Tool

The Directory Proxy Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the `collect-support-data` tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool varies between systems. However, the tool always tries to get the same information across all systems for the target Directory Proxy Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Available Tool Options

The `collect-support-data` tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.
- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.
- **--sequential**. Use this option to diagnose “Out of Memory” errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.

- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, `vmstat`, `iostat`, or `mpstat`). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.
- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, `mpstat`). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.
- **--maxJstacks {number}**. Specifies the number of jstack samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they will impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. The comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain `dsreplication status` information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain `dsreplication status` information.

To Run the Collect Support Data Tool

1. Go to the server root directory.
2. Use the `collect-support-data` tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --serverRoot /opt/PingDirectoryProxy --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Directory Proxy Server Troubleshooting Tools

The PingDirectoryProxy Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server Version Information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Proxy Server software that is in use. If the server is running, then this information can be obtained from the "`cn=Version,cn=monitor`" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.
- Source revision number from which the server was built.
- Build information including build ID with time stamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Jetty, JZlib, SNMP4j (SNMP4J, Agent, Agentx), Groovy, LDAP SDK for Java, and the Server SDK.

LDIF Connection Handler

The Directory Proxy Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This Directory Proxy Server does not exist by default, but if it is created and an LDIF file is placed in it that contains one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

Embedded Profiler

If the Directory Proxy Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the `dsconfig` tool or the Administrative Console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The `profile-action` property for this configuration object can have one of the following values:

- **start** – Indicates that the embedded profiler should start capturing data in the background.
- **stop** – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a `logs/profile{timestamp}` file.
- **cancel** – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the `profiler-viewer` tool. This tool can operate in either a text-based mode, in which case it dumps a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

To Invoke the Profile Viewer in Text-based Mode

- Run the `profile-viewer` command and specify the captured log file using the --fileName option.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

To Invoke the Profile Viewer in GUI Mode

- Run the `profile-viewer` command and specify the captured log file using the --fileName option. To invoke GUI mode, add the option --useGUI.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Troubleshooting Resources for Java Applications

Because the PingDirectoryProxy Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Proxy Server. In many cases, obtaining the full benefit of these tools requires access to the Directory Proxy Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java Troubleshooting Tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the PingDirectoryProxy Server.

jps

The `jps` tool is a Java-specific version of the UNIX `ps` tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Proxy Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- **-v** – Includes the arguments passed to the JVM for the processes that are listed.
- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

jstack

The `jstack` tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can have user-friendly names (as do the threads used by the PingDirectoryProxy Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where {processID} is the process ID of the target JVM as returned by the `jps` command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the java command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The "-l" option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The “-m” option can be used to include native frames in the stack trace.

jmap

The `jmap` tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native `pmap` tool provided by many operating systems. As with the `jstack` tool, `jmap` can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the `jhat` tool

- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.
- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the ":live" portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

jhat

The `jhat` (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory Proxy Server encounters an out of memory error (as a result of the "-XX:+HeapDumpOnOutOfMemoryError" JVM option) or from the use of the `jmap` command with the "-dump" option.

The `jhat` tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The `jhat` tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the `jhat` web server to begin listening on port 7000. It can be accessed in a browser at `http://localhost:7000` (or `http://address:7000` from a remote system). An alternate port number can be specified using the "-port" option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL `http://localhost:7000/oql/` (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at `http://localhost:7000/oqlhelp/`.

jstat

The `jstat` tool is used to obtain a variety of statistical information from the JVM, much like the `vmstat` utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The {interval} option specifies the length of time in milliseconds between lines of output. The {processID} option specifies the process ID of the JVM used to run the Directory Proxy Server, which can be obtained by running `jps` as mentioned previously. The {type} option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.
- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.
- **-gccapacity** – Provides information about memory region capacities.

Java Diagnostic Information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the `jstat` utility with the -gc option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

To run the Directory Proxy Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`bin/start-server.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

JVM Crash Diagnostic Information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the base directory of the Directory Proxy Server installation. This file includes information on the underlying cause of the JVM crash, information about the threads running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system.

Troubleshooting Resources in the Operating System

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Proxy Server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the Directory Proxy Server, and in others system, tools can help narrow down the cause of the problem.

Identifying Problems with the Underlying System

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. To look for problems in the underlying system view the system log file (`/var/log/messages` on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.

Monitoring System Data Using the PingDataMetrics Server

The PingDataMetrics Server provides collection and storage of performance data from your server topology. You can use the System Utilization Monitor with the PingDataMetrics Server to collect information about the host system CPU, disk, and network utilization on any platform except Linux. If you are not using the PingDataMetrics Server, you do not need to use the system utilization monitor. When data is being collected, it periodically forks the process and executes commands.

For more information about using the System Utilization Monitor, refer to the data collection chapter of the PingDataMetrics Server documentation.

Examining CPU Utilization

Observing CPU utilization for the Directory Proxy Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU Utilization

To investigate CPU consumption of the system as a whole, use the `vmstat` command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Proxy Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Proxy Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU Utilization

To investigate CPU consumption on a per-CPU basis, use the `mpstat` command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "-P ALL" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but `mpstat` reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-Process Utilization

To investigate CPU consumption on a per-process basis, use a command such as the `top` utility on Linux. If a process other than the Java process used to run the Directory Proxy Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Proxy Server to run effectively.

Examining Disk Utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Proxy Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The `iostat` tool may be used to obtain information about the disk activity on the system.

On Linux systems, `iostat` should be invoked with the "-x" argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but to obtain an initial feel for how busy the underlying disks are, look at the "%util" column on Linux. This field shows the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Proxy Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a `tmpfs` filesystem).

Examining Process Details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard `ps` tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard `ps` command displays only a limited set of the arguments based on available space in the terminal window.

pstack

The `pstack` command can be used to obtain a native stack trace of all threads in a process. While a native stack trace might not be as user-friendly as a Java stack trace obtained using `jstack`, it includes threads that are not available in a Java stack trace. For example, the command displays those threads used to perform garbage collection and other housekeeping tasks. The general usage for the `pstack` command is:

```
pstack {processID}
```

dbx / gdb

A process debugger provides the ability to examine a process in detail. Like `pstack`, a debugger can obtain a stack trace for all threads in the process, but it also provides the ability to examine a process (or core file) in much greater detail, including observing the contents of memory at a specified address and the values of CPU registers in different frames of execution. The GNU debugger `gdb` is widely-used on Linux systems.

Note that using a debugger against a live process interrupts that process and suspends its execution until it detaches from the process. In addition, when running against a live process, a debugger has the ability to actually alter the contents of the memory associated with that process, which can have adverse effects. As a result, it is recommended that the use of a process debugger be restricted to core files and only used to examine live processes under the direction of your authorized support provider.

pfiles / lsof

To examine the set of files that a process is using (including special types of files, like sockets), you can use a tool such as `lsof` on Linux systems, (

```
lsof -p {processID}
```

)

Tracing Process Execution

If a process is unresponsive but is consuming a nontrivial amount of CPU time, or if a process is consuming significantly more CPU time than is expected, it might be useful to examine the activity of that process in more detail than can be obtained using a point-in-time snapshot. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

This can be accomplished using the `strace` tool on Linux (

```
strace -f -p {processID}
```

).

Consult the `strace` manual page for additional information.

Problems with SSL Communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-server` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Examining Network Communication

Because the PingDirectoryProxy Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Proxy Server itself can provide details about its interaction with clients by enabling debugging for the protocol or data debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the *tcpdump* tool on Linux, can be used to accomplish this.

There are many options that can be used with these tools, and their corresponding manual pages will provide a more thorough explanation of their use. However, to perform basic tracing to show the full details of the packets received for communication on port 389 with remote host 1.2.3.4, the following command can be used on Linux:

```

tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

It does not appear that the `tcpdump` tool provides support for LDAP parsing. However, it is possible to write capture data to a file rather than displaying information on the terminal (using "-w {path}" with `tcpdump`), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access.

Common Problems and Potential Solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General Methodology to Troubleshoot a Problem

When a problem is detected, Ping Identity recommends using the following general methodology to isolate the problem:

1. Run the `bin/status` tool or look at the server status in the Administrative Console. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts.
2. Look in the server logs. In particular, view the following logs:

   logs/errors
   logs/failed-ops
   logs/expensive-ops
3. Use system commands, such as `vmstat` and `iostat` to determine if the server is bottle-necked on a system resource like CPU or disk throughput.
4. For performance problem (especially intermittent ones like spikes in response time), enabling the `periodic-stats-logger` can help to isolate problems, because it stores important server performance information on a per-second basis. The `periodic-stats-logger` can save the information in a csv-formatted file that can be loaded into a spreadsheet. The information this logger makes available is very configurable. You can create multiple loggers for different types of information or a different frequency of logging (for example, hourly data in addition to per-second data). For more information, see "Profiling Server Performance Using the Periodic Stats Logger".

5. For replication problem, run `dsreplication status` and look at the `logs/replication` file.
6. For more advanced users, run the `collect-support-data` tool on the system, unzip the archive somewhere, and look through the collected information. This is often useful when administrators most familiar with the Ping Identity Platform do not have direct access to the systems where the production servers are running. They can examine the `collect-support-data` archive on a different server. For more information, see Using the Collect Support Data Tool.

Important: Run the `collect-support-data` tool whenever there is a problem whose cause is not easily identified, so that this information can be passed back to your authorized support provider before corrective action can be taken.

The Server Will Not Run Setup

If the `setup` tool does not run properly, some of the most common reasons include the following:

A Suitable Java Environment Is Not Available

The PingDirectoryProxy Server requires that Java be installed on the system and made available to the server, and it must be installed prior to running `setup`. If the `setup` tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the `setup` tool should be invoked with an explicitly-defined value for the *JAVA_HOME* environment variable that specifies the path to the Java installation that should be used. For example:

```
env JAVA_HOME=/ds/java ./setup
```

If this still does not work for some reason, then it can be that the value specified in the provided *JAVA_HOME* environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected Arguments Provided to the JVM

If the `setup` script attempts to launch the java command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running setup, but this might not be the case if either the *JAVA_ARGS* or *UNBOUNDID_JAVA_ARGS* environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run `setup`:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Used

The `setup` tool is only intended to provide the initial configuration for the Directory Proxy Server. It refuses to run if it detects that the `setup` tool has already been run, or if an attempt has been made to start the Directory Proxy Server prior to running the `setup` tool. This protects an existing Directory Proxy Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Proxy Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run `setup` in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run `setup` failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the `setup` program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The Server Will Not Start

If the Directory Proxy Server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running

Only a single instance of the Directory Proxy Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other administrative operations can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Proxy Server could not acquire an exclusive lock on file
/ds/PingDirectoryProxy/locks/server.lock: The exclusive lock requested for
 file
/ds/PingDirectoryProxy/locks/ server.lock was not granted, which indicates
that another process already holds a shared or exclusive lock on that
file. This generally means that another instance of this server is already
running
```

If the Directory Proxy Server is not running (and is not in the process of starting up or shutting down) and there are no other tools running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the `locks` directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the filesystem.

There Is Not Enough Memory Available

When the Directory Proxy Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Proxy Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Proxy Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Proxy Server might need to be re-configured to use a smaller amount of memory than had been previously configured.
- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Proxy Server, or reconfigure the Directory Proxy Server to reduce the amount of memory that it attempts to use.
- The Directory Proxy Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the `vmstat` command and wait until the amount of free memory stops growing before attempting to restart the server.
- If the system is configured with one or more memory-backed filesystems, then look to see if there are any large files that can be consuming a significant amount of memory in any of those locations. If so, then remove them or relocate them to a disk-based filesystem.
- For Linux systems only, if there is a mismatch between the huge pages setting for the JVM and the huge pages reserved in the operating system.

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An Invalid Java Environment or JVM Option Was Used

If an attempt to start the Directory Proxy Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the `config/java.properties` file to reference to path to the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.
- If an *UNBOUNDID_JAVA_HOME* or *UNBOUNDID_JAVA_BIN* environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the `config/java.properties` file. Similarly, if an *UNBOUNDID_JAVA_ARGS* environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the *UNBOUNDID_JAVA_HOME*, *UNBOUNDID_JAVA_BIN*, and *UNBOUNDID_JAVA_ARGS* environment variables before trying to start the server.

Note that any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, then it can be necessary to remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option Was Provided

There are a small number of arguments that are provided when running the `bin/start-server` command, but in most cases, none are required. If one or more command-line arguments were provided for the `bin/start-server` command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration

If a change is made to the Directory Proxy Server configuration using an officially-supported tool like `dsconfig` or the Administrative Console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Proxy Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the `logs/config-audit.log` file to see what recent configuration changes have been made with the server online, or in the `config/archived-configs` directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the --useLastKnownGoodConfig option:

```
$ bin/start-server --useLastKnownGoodConfig
```

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the `config/config.ldif` file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.

Note: In addition to manually-editing the config file, you can look at previous achived configurations to see if the most recent one works. You can also use the `ldif-diff` tool to compare the configurations in the archive to the current configuration to see what is different.

You Do Not Have Sufficient Permissions

The Directory Proxy Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Proxy Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Proxy Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/PingDirectoryProxy
```

The Server Has Crashed or Shut Itself Down

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Proxy Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using “kill -9”), then messages are written to the `error` and `server.out` logs explaining the reason for the shutdown.
- The Directory Proxy Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the `uptime` command on the underlying system shows that it was recently booted.
- The Directory Proxy Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Proxy Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the `error` and `server.out` logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Proxy Server was running crashed. If this happens, then the JVM should dump a fatal error log (a `hs_err_pid{processID}.log` file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

Conditions for Automatic Server Shutdown

All PingDirectoryProxy Server servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The Directory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
           --set unrecoverable-database-error-mode:initiate-server-shutdown
```

The Server Will Not Accept Client Connections

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Proxy Server is not running.
- The underlying system on which the Directory Proxy Server is installed is not running.
- The Directory Proxy Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.
- If the Directory Proxy Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.
- If the Directory Proxy Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Proxy Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The Server is Unresponsive

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the `jstack` command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

  A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

  Note: If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.
- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the `jstack` command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.
- If the JVM in which the Directory Proxy Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a `pstack` of the native process should show that most threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will

also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.

- If the JVM in which the Directory Proxy Server is running has hung for some reason, then the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like `snoop` or `tcpdump` should show that packets sent to the system on which the Directory Proxy Server is running are not receiving TCP acknowledgement.
- If the system on which the Directory Proxy Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Proxy Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is Slow to Respond to Client Requests

If the Directory Proxy Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Proxy Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see "Profiling Server Performance Using the Periodic Stats Logger". The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second. You can also run the `bin/status` command or view the status in the Administrative Console to see the Directory Proxy Server's Work Queue information (also see the next bullet point).
- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Proxy Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the `bin/status` command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

  ```
             --- Work Queue ---
             : Recent : Average : Maximum
  -----------:--------:---------:--------
  Queue Size : 10   : 1       : 10
  % Busy     : 17     : 14      : 100
  ```

  You can also view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second.
- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, `iostat` reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see Disk-Bound Deployments.
- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.
- If the Directory Proxy Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, `iostat`

should report very high utilization for disks used to hold swap space, and commands like `cat /proc/meminfo` on Linux can report a large amount of swap memory in use. Another cause of swapping is if swappiness is not set to 0 on Linux. For more information, see Disable File System Swapping.

- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Proxy Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server Returns Error Responses to Client Requests

If a large number of client requests are receiving error responses, then view the `logs/failed-ops` log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.
- If a portion of the Directory Proxy Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxies or load balancers or both to route requests away from the affected server. As of Directory Proxy Server version 3.1 or later, the Directory Proxy Server will indicate that it is in a degraded status and the Directory Proxy Server will route around it.
- If the Directory Proxy Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.
- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

The Server Must Disconnect a Client Connection

If a client connection must be disconnected due to the expense of the client's request, such as an unindexed search across a very large database, perform the following:

- Find the client's connection ID by looking in the `cn=Active Operations,cn=monitor monitor` entry.

```
$ bin/ldapsearch -baseDN cn=monitor "cn=active operations" \
  --bindDN "cn=directory manager"  \
  --bindPassword password
```

- The monitor entry will contain attribute values for `operation-in-progress`, which look like an access log message. Look for the value of `conn` in the client request that should be disconnected. In the following example, the client to be disconnected is requesting a search for `(description=expensive)`, which is on connection 6.

```
dn: cn=Active Operations,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-active-operations-monitor-entry
objectClass: extensibleObject
cn: Active Operations
num-operations-in-progress: 2
operation-in-progress: [15/Dec/2014:10:55:35 -0600] SEARCH conn=6 op=3
 msgID=4
```

```
    clientIP="10.8.4.21" authDN="cn=app1,ou=applications,dc=example,dc=com"
 base="dc
    =example,dc=com" scope=wholeSubtree filter="(description=expensive)"
 attrs="A
    LL" unindexed=true
operation-in-progress: [15/Dec/2014:10:56:11 -0600] SEARCH conn=7 op=1
 msgID=2
    clientIP="127.0.0.1" authDN="cn=Directory Manager,cn=Root
 DNs,cn=config" base="c
    n=monitor" scope=wholeSubtree filter="(cn=active operations)"
 attrs="ALL"
    num-persistent-searches-in-progress: 0
```

- With the connection ID value, create a file with the following contents, named `disconnect6.ldif`.

```
dn: ds-task-id=disconnect6,cn=scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-disconnect
ds-task-disconnect-connection-id: 6
ds-task-id: disconnect6
ds-task-class-name:
 com.unboundid.directory.server.tasks.DisconnectClientTask
```

- This LDIF file represents a task entry. The connection ID value 6 is assigned to `ds-task-disconnect-connection-id`. The value for `ds-task-id` value does not follow a specific convention. It must be unique among other task entries currently cached by the server.
- Disconnect the client and cancel the associated operation by adding the task entry to the server:

```
$ bin/ldapmodify --filename disconnect6.ldif  \
  --defaultAdd --bindDN "cn=directory manager" \
  --bindPassword password
```

Problems with the Administrative Console

If a problem arises when trying to use the Administrative Console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Proxy Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with the Directory Proxy Server instance using the Administrative Console, then the access and error logs for that Directory Proxy Server instance might provide additional information about the underlying problem.

Problems with the Administrative Console: JVM Memory Issues

Console runs out of memory (PermGen). An inadequate PermSize setting in the server, while hosting web applications like the Administrative Console may result in errors like this in the error log:

```
[02/Mar/2016:07:50:27.017 -0600] threadID=2 category=UTIL
    severity=SEVERE_ERROR msgID=-1 msg="The server experienced an unexpected
    error. Please report this problem and include this log file.
    OutOfMemoryError: PermGen space
    ()\ncom.unboundid.directory.server.core.DirectoryServer.uncaughtException
    (DirectoryServer.java:15783)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1057)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.Thread.dispatchUncaughtException
    (Thread.java:1986)\nBuild revision: 22496\n"
```

This is only relevant for servers running Java 7.

Global Index Growing Too Large

If the global index appears to be growing too large, you can reload from the backend directory servers. Use the `reload-index` tool with the --fromDS option, overriding the configuration of the `prime-index-source` property. You can do this on a one off basis if the global index appears to be growing too large as follows:

```
$ bin/reload-index \
  --bindPassword password \
  --baseDN "dc=example,dc=com" \
  --fromDS
```

Forgotten Proxy User Password

If you have forgotten the password you set for the `cn=Proxy User` entry, you can work around the problem as follows:

- You can temporarily add a second password to the proxy user entry so that you can transition all of the proxy server instances to the new password. However, you should have multiple passwords on the `cn=Proxy User` entry for the shortest time possible.
- If you do not know the clear-text value, then you can use the encrypted value when configuring the new Directory Proxy Server. The encryption scheme allows reversible passwords that are stored in the server configuration so that they can be decrypted by any server instance.
- You can create a new root user in the directory server instances with the appropriate set of privileges and have the new proxy server instance use that account to authenticate. Since it is not a good idea to have an account for which you do not know the password, you may want to update all of the other proxy server instances to use the new account.
- You can use a protocol analyzer like `snoop` or `Wireshark`, to capture the password from the network communication.

Providing Information for Support Cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the `collect-support-data` tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

Chapter 11

Managing the SCIM Servlet Extension

Topics:

The PingDirectoryProxy Server provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS.

This section describes fundamental SCIM concepts and provides information on configuring SCIM on your server.

Overview of SCIM Fundamentals

Understanding the basic concepts of SCIM can help you use the SCIM extension to meet the your deployment needs. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension implements the 1.1 version of the SCIM specification. Familiarize yourself with this specification to help you understand and make efficient use of the SCIM extension and the SCIM SDK. The SCIM specifications are located on the Simplecloud website.

Summary of SCIM Protocol Support

PingDirectoryProxy Server implements all required features of the SCIM protocol and most optional features. The following table describes SCIM features and whether they are supported.

Table 12: SCIM Protocol Support

SCIM Feature	Supported
Etags	Yes
JSON	Yes
XML*	Yes
Authentication/Authorization	Yes, via HTTP basic authentication or OAuth 2.0 bearer tokens
Service Provider Configuration	Yes
Schema	Yes
User resources	Yes
Group resources	Yes
User-defined resources	Yes
Resource retrieval via GET	Yes
List/query resources	Yes
Query filtering*	Yes
Query result sorting*	Yes
Query result pagination*	Yes (Directory Server, not Directory Proxy Server)
Resource updates via PUT	Yes
Partial resource updates via PATCH*	Yes
Resource deletes via DELETE	Yes
Resource versioning*	Yes (requires configuration for updated servers)
Bulk*	Yes
HTTP method overloading	Yes
Raw LDAP Endpoints**	Yes

* denotes an optional feature of the SCIM protocol.

** denotes a PingDirectoryProxy Server extension to the basic SCIM functionality.

About the Identity Access API

The PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server support an extension to the SCIM standard called the Identity Access API. The Identity Access API provides an alternative to LDAP by supporting CRUD (create, read, update, and delete) operations to access directory server data over an HTTP connection.

SCIM and the Identity Access API are provided as a unified service through the SCIM HTTP Servlet Extension. The SCIM HTTP Servlet Extension can be configured to only enable core SCIM resources (e.g., 'Users' and 'Groups'), only LDAP object classes (e.g., `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there may be two representations with different schemas for any resources defined in the `scim-resources.xml` file: the SCIM representation and the raw LDAP representation. Likewise, because resources are exposed by an LDAP object class, and because these are hierarchical (e.g., `top` --> `person` --> `organizationalPerson` --> `inetOrgPerson`, etc.), a client application can access an entry in multiple ways due to the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the PingDirectory Server.

Creating Your Own SCIM Application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM SDK is available for download at https://github.com/pingidentity/scim.

Note: The value of a read-only SCIM attribute can be set by a POST operation if the SCIM attribute is a custom attribute in the `scim-resource.xml` config file, but not if the SCIM attribute is a core SCIM attribute.

Configuring SCIM

This section discusses details about the PingDirectoryProxy Server implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available on the Simplecloud website.

Before You Begin

To set up your SCIM servlet extension, the Directory Server provides a `dsconfig` batch file file, `scim-config-proxy.dsconfig`, located in the `<server-root>/config` directory. The script runs a series of commands that enables the HTTP Connection Handler and SCIM HTTP Servlet Extension, increases the level of detail logged by the HTTP Detailed Access Log Publisher, adds access controls to allow access to LDAP controls used by the SCIM servlet, adds support to the request processor for LDAP controls used by the SCIM servlet, and sets the subordinate base DN property of the root DSE so that SCIM requests can be authenticated using LDAP `uid` values. You should edit this `dsconfig` batch file before running the details of your deployment.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See *Managing the SCIM Schema* for more information.

Configuring the SCIM Servlet Extension

The Directory Proxy Server provides a default SCIM HTTP Servlet Extension that can be enabled and configured using a `dsconfig` batch script, `scim-config-proxy.dsconfig`, located in the `config` directory. The script runs a series of commands that enables the HTTPS Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension.

When configuring the Directory Proxy Server to act as a SCIM server, enable the `entryDN` virtual attribute on any directory servers fronted by the Directory Proxy Server. This is also needed when using the Identity Access API.

To Configure the SCIM Servlet Extension

1. Before you enable the SCIM servlet extension, add access controls on each of the backend Directory Servers to allow read access to operational attributes used by the SCIM Servlet Extension. We recommend using the following non-interactive command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id
 ||
    createTimestamp || modifyTimestamp")
    (version 3.0;acl "Authenticated read access to operational attributes \
    used by the SCIM servlet extension"; allow (read,search,compare)
    userdn="ldap:///all";)'
```

2. On the Directory Proxy Server, enable the SCIM servlet extension by running the `dsconfig` batch file.

```
$ bin/dsconfig --batch-file config/scim-config-proxy.dsconfig
```

3. The `dsconfig` batch file must be edited to use the correct request processor name and base DN name(s) for the `set-request-processor-prop` and `set-root-dse-backend-prop` commands, respectively, as described in the "Configuring LDAP Control Support on All Request Processors" and "SCIM Servlet Extension Authentication" sections later in the chapter.

To Enable Resource Versioning

Resource versioning is enabled by default in new installations. Upgraded servers that had SCIM enabled need additional configuration to enable resource versioning.

1. Enable the `ds-entry-checksum` virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
                        --name ds-entry-checksum \
                        --set enabled:true
```

2. Remove any existing access controls required by SCIM for read access to operational attributes:

```
$ bin/dsconfig set-access-control-handler-prop \
                        --remove 'global-aci:(targetattr="entryUUID ||
 entryDN || ds-entry-unique-id || createTimestamp || ds-create-time ||
 modifyTimestamp || ds-update-time")(version 3.0;acl "Authenticated read
 access to operational attributes used by the SCIM servlet extension"; allow
 (read,search,compare) userdn="ldap:///all"'
```

3. On the backend Directory Server, make sure new access controls required by SCIM for read access to operational attributes are enabled with the following command. If this ACI is not present, issues will occur when a SCIM client tries to authenticate with an non-root DN.

```
$ bin/dsconfig set-access-control-handler-prop \
                        --add  'global-aci:(targetattr="entryUUID ||
 entryDN || ds-entry-unique-id || createTimestamp || ds-create-time ||
 modifyTimestamp || ds-update-time || ds-entry-checksum")(version 3.0;acl
 "Authenticated read access to operational attributes used by the SCIM
 servlet extension"; allow (read,search,compare) userdn="ldap:///all"'
```

Configuring LDAP Control Support on All Request Processors (Proxy Only)

You need to configure support for the required LDAP controls on all request processors handling LDAP requests that result from SCIM requests. Change the request processor name that was provided as an example and repeat the command for all additional request processors.

To Configure LDAP Control Support on All Request Processors

- Use `dsconfig` to change the request processor name that was provided as an example and repeat the command for all additional request processors. Make sure to use your deployment's request processor name.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add supported-control-oid:1.2.840.113556.1.4.319 \
  --add supported-control-oid:1.2.840.113556.1.4.473 \
  --add supported-control-oid:2.16.840.1.113730.3.4.9
```

SCIM Servlet Extension Authentication

The SCIM servlet supports authentication using either the HTTP Basic authentication scheme, or OAuth 2.0 bearer tokens. When authenticating using HTTP Basic authentication, the SCIM servlet attempts to correlate the username component of the Authorization header to a DN in the Directory Proxy Server. If the username value cannot be parsed directly as a DN, it is correlated to a DN using an Identity Mapper. The DN is then used in a simple bind request to verify the password.

In deployments that use an OAuth authorization server, the SCIM extension can be configured to authenticate requests using OAuth bearer tokens. The SCIM extension supports authentication with OAuth 2.0 bearer tokens (per RFC 6750) using an OAuth Token Handler Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, PingDirectoryProxy Server provides the token handler API to decode incoming bearer tokens and extract or correlate associated authorization DNs.

Neither HTTP Basic authentication nor OAuth 2.0 bearer token authentication are secure unless SSL is used to encrypt the HTTP traffic.

Enabling HTTPS Communications

If you want the SCIM HTTP connection handler to use SSL, which is mandated by the SCIM specification, you need to enable a Key Manager provider and Trust Manager provider.

To enable SSL during the Directory Proxy Server's initial setup, include the --ldapsPort and the --generateSelfSignedCertificate arguments with the `setup` command. If your server already has a certificate that you would like to use, set the `key-manager-provider` to the value you set when you enabled SSL in the Directory Proxy Server, or define a new key manager provider (see Configuring HTTP Connection Handlers).

To Configure Basic Authentication Using an Identity Mapper

By default, the SCIM servlet is configured to use the Exact Match Identity Mapper, which matches against the `uid` attribute. In this example, an alternate Identity Mapper is created so that clients can authenticate using `cn` values.

1. Create a new Identity Mapper that uses a match attribute of cn.

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "CN Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:cn
```

2. Configure the SCIM servlet to use the new Identity Mapper.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set "identity-mapper:CN Identity Mapper"
```

To Enable OAuth Authentication

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the Server SDK documentation.

1. Install your OAuth token handler on the server using `dsconfig`.

   ```
   $ bin/dsconfig create-oauth-token-handler \
     --handler-name ExampleOAuthTokenHandler \
     --type third-party \
     --set extension-
   class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
   ```
2. Configure the SCIM servlet extension to use it as follows:

   ```
   $ bin/dsconfig set-http-servlet-extension-prop \
     --extension-name SCIM \
     --set oauth-token-handler:ExampleOAuthTokenHandler
   ```

Using HTTP Basic Authentication with Bare UID on the Directory Proxy Server

As discussed above, clients can authenticate to the SCIM extension using HTTP basic authentication and a bare UID value. However, when a SCIM extension is hosted by a Directory Proxy Server, the server needs to be explicitly configured with the names of subordinate base DNs to search. To do this, run the following command on the Directory Proxy Server for every base DN that may be accessed via SCIM. Make sure to specify your deployment's subordinate base DN.

```
$ bin/dsconfig set-root-dse-backend-prop \
  --set subordinate-base-dn:dc=example,dc=com
```

Verifying the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

To Verify the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

- Run `curl` to verify that the SCIM extension is running. The -k (or --insecure) option is used to turn off curl's verification of the server certificate, since the example Directory Proxy Server is using a self-signed certificate.

  ```
  $ curl -u "cn=Directory Manager:password" \
  -k "https://localhost:8443/scim/ServiceProviderConfigs"

  {"schemas":["urn:scim:schemas:core:1.0"],"id":"urn:scim:schemas:core:1.0",
  "patch":{"supported":true},"bulk":{"supported":true,"maxOperations":10000,
  "maxPayloadSize":10485760},"filter":{"supported":true,"maxResults":100},
  "changePassword":{"supported":true},"sort":{"supported":true},
  "etag":{"supported":false},"authenticationSchemes":[{"name":"HttpBasic",
  "description":"The HTTP Basic Access Authentication scheme. This scheme is
  not considered to be a secure method of user authentication (unless used in
  conjunction with some external secure system such as SSL), as the user
  name and password are passed over the network as cleartext.","specUrl":
  "http://www.ietf.org/rfc/rfc2617","documentationUrl":
  "http://en.wikipedia.org/wiki/Basic_access_authentication"}]}
  ```
- If the user ID is a valid DN (such as `cn=Directory Manager`), the SCIM extension authenticates by binding to the Directory Proxy Server as that user. If the user ID is not a valid DN, the SCIM extension searches for an entry with that `uid` value, and binds to the server as that user. To verify authentication to the server as the user with the `uid` of `user.0`, run the following command:

```
$ curl -u "user.0:password" \
  -k "https://localhost:8443/scim/ServiceProviderConfigs"
```

Configuring Advanced SCIM Extension Features

The following sections show how to configure advanced SCIM servlet extension features, such as bulk operation implementation, mapping SCIM resource IDs, and transformations.

Managing the SCIM Schema

This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About SCIM Schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping LDAP Schema to SCIM Resource Schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Proxy Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

Note that LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). The easiest and most-correct way to handle this is to make sure that any attributes that may contain binary data are declared using "dataType=binary" in the `scim-resources.xml` file. Likewise, when using the Identity Access API make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes which may contain binary data. This will cause the server to automatically base64-encode the data before returning it to clients and will also make it predictable for clients because they can assume the data will always be base64-encoded.

However, it is still possible that attributes that are not declared as binary in the schema may contain binary data (or just data that is invalid in XML), and the server will always check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values which are found to include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the <resource> Element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A resource element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the <attribute> Element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as urn:scim:schemas:core:1.0. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is false.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:

  `simple`: specifies a simple, singular SCIM attribute.
  `complex`: specifies a complex, singular SCIM attribute.
  `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
  `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the <simple> Element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: binary, boolean, dateTime, decimal, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simple element contains the following XML element:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the <complex> Element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

About the <simpleMultivalued> Element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simpleMultiValued element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the <complexMultiValued> Element

A `complexMultiValued` element has the following XML attribute:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is address.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.
- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the <subAttribute> Element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: binary, boolean, dateTime, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the complexMultiValued element.

About the <canonicalValue> Element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the <mapping> Element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in the *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API* section.

About the <subMapping> Element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. The available transformations are described in *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API*.

About the <LDAPSearch> Element

An `LDAPSearch` element contains the following XML elements in sequence:

- `baseDN`: a required element specifying one or more LDAP search base DNs to be used when querying for the SCIM resource.
- `filter`: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- `resourceIDMapping`: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN. Note The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

Note: The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

About the <resourceIDMapping> Element

The `resourceIDMapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `scim-consumer`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or Directory Proxy Server, meaning that a value is automatically provided by the Directory Proxy Server (as would be the case if the mapped LDAP attribute is entryUUID).

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the <LDAPAdd> Element

An LDAPAdd element contains the following XML elements in sequence:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

About the <fixedAttribute> Element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The value merge indicates that the fixed values should be merged with the existing values. The value overwrite indicates that the existing values are to be overwritten by the fixed values. The value preserve indicates that no changes should be made. The default value is merge.

A `fixedAttribute` element contains one or more `fixedValue` XML element, which specify the fixed LDAP values.

Validating Updated SCIM Schema

The PingDirectoryProxy Server SCIM extension is bundled with an XML Schema document, `resources.xsd`, which describes the structure of a `scim-resources.xml` resource configuration file. After updating the resource configuration file, you should confirm that its contents are well-formed and valid using a tool such as `xmllint`.

For example, you could validate your updated file as follows:

```
$ xmllint --noout --schema resources.xsd scim-resources.xml
scim-resources.xml validates
```

Mapping SCIM Resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. The `entryUUID` attribute, whose read-only value is assigned by the Directory Proxy Server, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership. The resource configuration allows the SCIM resource ID to be mapped to the LDAP entry DN. However, the entry DN does not meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN. Likewise, you can use the Identity Access API to change the value of an entry's RDN attribute, thereby triggering a MODDN operation.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Directory Proxy Server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin may be used by the Directory Proxy Server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see "Working with the UID Unique Attribute Plug-in" in the PingDirectory Server Administration Guide.

Note: Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the `<resourceIDMapping>` Element".

Using Pre-defined Transformations

Transformations are required to change SCIM data types to LDAP syntax values. The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation`. Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation`. Transforms SCIM dateTime data type values to LDAP Generalized Time syntax values and vice-versa.
- `com.unboundid.scim.ldap.PostalAddressTransformation`. Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the $ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.
- `com.unboundid.scim.ldap.TelephoneNumberTransformation`. Transforms LDAP Telephone Number syntax (E.123) to RFC3966 format and vice-versa.

You can also write your own transformations using the SCIM API described in the following section.

Mapping LDAP Entries to SCIM Using the SCIM-LDAP API

In addition to the SCIM SDK, PingDirectoryProxy Server provides a library called SCIM-LDAP, which provides facilities for writing custom transformations and more advanced mapping.

You can add the SCIM-LDAP library to your project using the following dependency:

```
<dependency>
    <groupId>com.unboundid.product.scim</groupId>
    <artifactId>scim-ldap</artifactId>
    <version>1.5.0</version>
</dependency>
```

Create your custom transformation by extending the `com.unboundid.scim.ldap.Transformation` class. Place your custom transformation class in a jar file in the server's `lib` directory.

Note: The Identity Access API automatically maps LDAP attribute syntaxes to the appropriate SCIM attribute types. For example, an LDAP DirectoryString is automatically mapped to a SCIM string.

SCIM Authentication

SCIM requests to the LDAP endpoints will support HTTP Basic Authentication and OAuth2 Authentication using a bearer token. There is existing support for this feature in the Directory Server and the Directory Proxy Server using the OAuthTokenHandler API (i.e., via a Server SDK extension, which requires some technical work to implement).

Note that our implementation only supports the HTTP Authorization header for this purpose; we do not support the form-encoded body parameter or URI query parameter mechanisms for specifying the credentials or bearer token.

SCIM Logging

The Directory Proxy Server already provides a detailed HTTP log publisher to capture the SCIM and HTTP request details. To be able to correlate this data to the internal LDAP operations that are invoked behind the scenes, the Access Log Publisher will use "`origin=scim`" in access log messages that are generated by the SCIM servlet.

For example, you will see a message for operations invoked by replication:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191
origin="replication" dn="uid=user.3,ou=people,dc=example,dc=com"
```

Likewise for SCIM messages, you will see a message like this:

```
[30/Oct/2012:18:45:10.490 -0500] MODFIY REQUEST conn=-3 op=190 msgID=191
origin="scim" dn="uid=user.3,ou=people,dc=example,dc=com"
```

SCIM Monitoring

There are two facilities that can be used to monitor the SCIM activity in the server.

- **HTTPConnectionHandlerStatisticsMonitorProvider** -- Provides statistics straight about total and average active connections, requests per connection, connection duration, processing time, invocation count, etc.
- **SCIMServletMonitorProvider** -- Provides high level statistics about request methods (POST, PUT, GET, etc.), content types (JSON, XML), and response codes, for example, "`user-patch-404:26`".

The LDAP object class endpoints are treated as their own resource types, so that for requests using the Identity Access API, there will be statistics, such as `person-get-200` and `inetorgperson-post-401`.

Configuring the Identity Access API

Once you have run the `<server-root>/config/scim-config-ds.dsconfig` script, the resources defined in the `scim-resources.xml` will be available as well as the Identity Access API. However, to allow SCIM access to the raw LDAP data, you must set a combination of configuration properties on the SCIM Servlet Extension using the `dsconfig` tool.

- **include-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will be exposed. The object class used here will be the one that clients need to use when referencing Identity Access API resources. This property allows the special value "*" to allow all object classes. If "*" is used, then the SCIM servlet uses the same case used in the Directory Proxy Server LDAP Schema.
- **exclude-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will not be exposed. When this property is specified, all object classes will be exposed except those in this list.
- **include-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will be exposed. If specified, only entries under these base DNs will be accessible. No parent-child relationships in the DNs are allowed here.
- **exclude-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will not be exposed. If specified, entries under these base DNs will not be accessible. No parent-child relationships in the DNs are allowed here.

Using a combination of these properties, SCIM endpoints will be available for all included object classes, just as if they were SCIM Resources defined in the `scim-resources.xml` file.

To Configure the Identity Access API

1. Ensure that you have run the `scim-config-ds.dsconfig` script to configure the SCIM interface. Be sure to enable the entryDN virtual attribute. See the Configure SCIM section for more information.
2. Set a combination of properties to allow the SCIM clients access to the raw LDAP data: `include-ldap-objectclass`, `exclude-ldap-objectclass`, `include-ldap-base-dn`, or `exclude-ldap-base-dn`.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM --set 'include-ldap-objectclass:*' \
  --set include-ldap-base-dn:ou=People,dc=example,dc=com
```

The SCIM clients now have access to the raw LDAP data via LDAP object class-based resources as well as core SCIM resources as defined in the `scim.resource.xml` file.

To Disable Core SCIM Resources

1. Open the `config/scim-resources.xml` file, and comment out or remove the `<resource>` elements that you would like to disable.
2. Disable and re-enable the HTTP Connection Handler, or restart the server to make the changes take effect. In general, changing the `scim-resources.xml` file requires a HTTP Connection Handler restart or server restart.

Note: When making other changes to the SCIM configuration by modifying the SCIM HTTP Servlet Extension using `dsconfig`, the changes take effect immediately without any restart required.

To Verify the Identity Access API Configuration

- Perform a curl request to verify the Identity Access API configuration.

```
$ curl -k -u "cn=directory manager:password" \
  -H "Accept: application/json" \
  "https://example.com/top/56c9fd6b-f870-35ef-9959-691c783b7318?
     attributes=entryDN,uid,givenName,sn,entryUUID"
     {"schemas":
["urn:scim:schemas:core:1.0","urn:unboundid:schemas:scim:ldap:1.0"],
      "id":"56c9fd6b-f870-35ef-9959-691c783b7318",
      "meta":{"lastModified":"2013-01-11T23:38:26.489Z",
      "location":"https://example.com:443/v1/top/56c9fd6b-
f870-35ef-9959-691c783b7318"},
      "urn:unboundid:schemas:scim:ldap:1.0":{"givenName":["Rufus"],"uid":
["user.1"],
      "sn":["Firefly"],"entryUUID":["56c9fd6b-f870-35ef-9959-691c783b7318"],
      "entrydn":"uid=user.1,ou=people,dc=example,dc=com"}}
```

Monitoring the SCIM Servlet Extension

The SCIM SDK provides a command-line tool, `scim-query-rate`, that measures the SCIM query performance for your extension. The SCIM extension also exposes monitoring information for each SCIM resource, such as the number of successful operations per request, the number of failed operations per request, the number of operations with XML or JSON to and from the client. Finally, the Directory Proxy Server automatically logs SCIM-initiated LDAP operations to the default File-based Access Logger. These operations will have an `origin='scim'` attribute to distinguish them from operations initiated by LDAP clients. You can also create custom logger or request criteria objects that can track incoming HTTP requests, which the SCIM extension rewrites as internal LDAP operations.

Testing SCIM Query Performance

You can use the `scim-query-rate` tool, provided in the SCIM SDK, to test query performance, by performing repeated resource queries against the SCIM server.

The `scim-query-rate` tool performs searches using a query filter or can request resources by ID. For example, you can test performance by using a filter to query randomly across a set of one million users with eight concurrent threads. The user resources returned to the client in this example is in XML format and includes the `userName` and `name` attributes.

```
scim-query-rate --hostname server.example.com --port 80 \
--authID admin --authPassword password --xml \
--filter 'userName eq "user.[1-1000000]"' --attribute userName \
--attribute name --numThreads 8
```

You can request resources by specifying a resource ID pattern using the --resourceID argument as follows:

```
scim-query-rate --hostname server.example.com --port 443 \
--authID admin --authPassword password --useSSL --trustAll\
--resourceName User \
--resourceID 'uid=user.[1-150000],ou=people,dc=example,dc=com'
```

The `scim-query-rate` tool reports the error "`java.net.SocketException: Too many open files`" if the open file limit is too low. You can increase the open file limit to increase the number of file descriptors.

Monitoring Resources Using the SCIM Extension

The monitor provider exposes the following information for each resource:

Number of successful operations per request type (such as GET, PUT, and POST).

Number of failed operations and their error codes per request type.

Number of operations with XML or JSON from client.

Number of operations that sent XML or JSON to client.

In addition to the information about the user-defined resources, monitoring information is also generated for the schema, service provider configuration, and monitor resources. The attributes of the monitor entry are formatted as follows:

```
{resource name}-resource-{request type}-{successful or error status code}
```

You can search for one of these monitor providers using an `ldapsearch` such as the following:

```
$ bin/ldapsearch --port 1389 bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN cn=monitor \
  --searchScope sub "(objectclass=scim-servlet-monitor-entry)"
```

For example, the following monitor output was produced by a test environment with three distinct SCIM servlet instances, Aleph, Beth, and Gimel. Note that the first instance has a custom resource type called host.

```
$ bin/ldapsearch --baseDN cn=monitor \
  '(objectClass=scim-servlet-monitor-entry)'
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Aleph)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Aleph)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
user-resource-delete-successful: 1
user-resource-put-content-xml: 27
user-resource-query-response-json: 3229836
user-resource-put-403: 5
user-resource-put-content-json: 2
user-resource-get-401: 1
user-resource-put-response-json: 23
user-resource-get-response-json: 5
user-resource-get-response-xml: 7
user-resource-put-400: 2
user-resource-query-401: 1141028
user-resource-post-content-json: 1
user-resource-put-successful: 22
user-resource-post-successful: 1
user-resource-delete-404: 1
user-resource-query-successful: 2088808
user-resource-get-successful: 10
user-resource-put-response-xml: 6
user-resource-get-404: 1
user-resource-delete-401: 1
user-resource-post-response-json: 1
```

```
host-resource-query-successful: 5773268
host-resource-query-response-json: 11576313
host-resource-query-400: 3
host-resource-query-response-xml: 5
host-resource-query-401: 5788152
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Beth)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Beth)
version: 1.2.0
build: 20120105174457Z
revision: 820
serviceproviderconfig-resource-get-successful: 3
serviceproviderconfig-resource-get-response-json: 2
serviceproviderconfig-resource-get-response-xml: 1
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
group-resource-query-successful: 245214
group-resource-query-response-json: 517841
group-resource-query-400: 13711
group-resource-query-401: 258916
user-resource-query-response-json: 107876
user-resource-query-400: 8288
user-resource-get-400: 33
user-resource-get-response-json: 1041
user-resource-get-successful: 2011
user-resource-query-successful: 45650
user-resource-get-response-xml: 1003
user-resource-query-401: 53938
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Gimel)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Gimel)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 1
schema-resource-query-401: 1
schema-resource-query-response-json: 2
user-resource-query-successful: 65
user-resource-get-successful: 4
user-resource-get-response-json: 6
user-resource-query-response-json: 132
user-resource-get-404: 2
user-resource-query-401: 67
```

About the HTTP Log Publishers

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.
- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.
- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as "'['dd/MM/yyyy:HH:mm:ss Z']'"
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information than the common log format in a format that is familiar to administrators who use the File-Based Access Log Publisher.

The HTTP Detailed Access Log Publisher generates log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-
38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-
d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the `dsconfig` command-line tool help.

Chapter

12

Managing Server SDK Extensions

Topics:

The PingDirectoryProxy Server provides support for any custom extensions that you create using the Server SDK. This chapter summarizes the various features and extensions that can be developed using the Server SDK.

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Proxy Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Available Types of Extensions

The Server SDK provides support for creating a number of different types of extensions for Ping Identity Server Products, including the PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. Some of those extensions include:

Cross-Product Extensions

Access Loggers
Alert Handlers
Error Loggers
Key Manager Providers
Monitor Providers
Trust Manager Providers
OAuth Token Handlers
Manage Extension Plugins

PingDirectory Server Extensions

Certificate Mappers
Change Subscription Handlers
Extended Operation Handlers
Identity Mappers
Password Generators
Password Storage Schemes
Password Validators
Plugins
Tasks
Virtual Attribute Providers

PingDirectoryProxy Server Extensions

LDAP Health Checks
Placement Algorithms
Proxy Transformations

PingDataSync Server Extensions

JDBC Sync Sources
JDBC Sync Destinations
LDAP Sync Source Plugins
LDAP Sync Destination Plugins
Sync SourcesSync Destinations

- Sync Pipe Plugins

For more information on the Server SDK, see the documentation available in the SDK build.

Chapter 13

Command-Line Tools

Topics:

The PingDirectoryProxy Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

This chapter presents the following topics:

Using the Help Option

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. You can view detailed argument options and examples by typing --help with the command.

```
bin/dsconfig --help
```

For those utilities that support additional subcommands (for example, `dsconfig`), you can get a list of the subcommands by typing --help-subcommands.

```
bin/dsconfig --help-subcommands
```

You can also get more detailed subcommand information by typing --help with the specific subcommand.

```
bin/dsconfig list-log-publishers --help
```

Note: For detailed information and examples of the command-line tools, see the *Ping Identity Directory Proxy Server Command-Line Tool Reference*.

Available Command-Line Utilities

The Directory Proxy Server provides the following command-line utilities, which can be run directly in interactive, non-interactive, or script mode.

Table 13: Command-Line Utilities

Command-Line Tools	Description
authrate	Perform repeated authentications against an LDAP directory server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more Directory Proxy Server backends. This utility also supports the use of a properties file to pass predefined command-line arguments. See *Managing the tools.properties File* for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a ZIP archive that can be sent to a technical support representative.
create-initial-proxy-config	Create an inital Directory Proxy Server configuration.
create-rc-script	Create an Run Control (RC) script that may be used to start, stop, and restart the Directory Proxy Server on UNIX-based systems.
create-recurring-task	Create a recurring task to run on the server. Tasks can be created for backups, LDIF exports, a statically defined task, or a third-party task.
create-recurring-task-chain	Create a chain of recurring tasks to run on the server.
dsconfig	View and edit the Directory Proxy Server configuration.
dsjavaproperties	Configure the JVM arguments used to run the Directory Proxy Server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and *JAVA_HOME* environment variable.

Command-Line Tools	Description
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Directory Server.
enter-lockdown-mode	Request that the Directory Proxy Server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege.
global-index-size	Estimates the size in memory of one or more global indexes from the actual number of keys, the configured number of keys and the average key size.
ldap-diff	Compare the contents of two LDAP directory server servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform LDAP compare operations in the Directory Proxy Server.
ldapdelete	Perform LDAP delete operations in the Directory Proxy Server.
ldapmodify	Perform LDAP modify, add, delete, and modify DN operations in the Directory Proxy Server.
ldappasswordmodify	Perform LDAP password modify operations in the Directory Proxy Server.
ldapsearch	Perform LDAP search operations in the Directory Proxy Server.
ldif-diff	Compare the contents of two LDIF files, the output being an LDIF file needed to bring the source file in sync with the target.
ldifmodify	Apply a set of modify, add, and delete operations against data in an LDIF file.
ldifsearch	Perform search operations against data in an LDIF file.
leave-lockdown-mode	Request that the Directory Proxy Server leave lockdown mode and resume normal operation.
list-backends	List the backends and base DNs configured in the Directory Proxy Server.
make-ldif	Generate LDIF data based on a definition in a template file.
manage-extension	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the PingDirectoryProxy Server. Extension bundles are installed from a zip archive or file system directory. The Directory Proxy Server will be restarted if running to activate the extension(s).
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Directory Proxy Server.
modrate	Perform repeated modifications against an LDAP directory server.
move-subtree	Move a subtree entries or a single entry from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
prepare-external-server	Prepare and a directory server for communication.
profile-viewer	View information in data files captured by the Directory Proxy Server profiler.
reload-index	Reload the contents of the global index.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Directory Proxy Server backend.
remove-defunct-server	Remove a server from this server's topology.
restore	Restore a backup of the Directory Proxy Server backend.

Command-Line Tools	Description
revert-update	Returns a server to the version before the last update was performed.
review-license	Review and/or indicate your acceptance of the product license.
scramble-ldif	Obscure the contents of a specified set of attributes in an LDIF file.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
search-rate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Directory Proxy Server process.
setup	Perform the initial setup for the Directory Proxy Server instance.
start-server	Start the Directory Proxy Server.
status	Display basic server information.
stop-server	Stop or restart the Directory Proxy Server.
subtree-accessibility	List or update the a set of subtree accessibility restrictions defined in the Directory Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
uninstall	Uninstall the Directory Proxy Server.
update	Update the Directory Proxy Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the `update` tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the `revert-update` tool for information on reverting an update.
validate-ldif	Validate the contents of an LDIF file against the server schema.

Managing the tools.properties File

The PingDirectoryProxy Server supports the use of a tools properties file that simplifies command-line invocations by reading in a set of arguments for each tool from a text file. Each property is in the form of name/value pairs that define predetermined values for a tool’s arguments. Properties files are convenient when quickly testing the Directory Proxy Server in multiple environments.

The Directory Proxy Server supports two types of properties file: default properties files that can be applied to all command-line utilities or tool-specific properties file that can be specified using the --propertiesFilePath option. You can override all of the Directory Proxy Server's command-line utilities with a properties file using the `config/tools.properties` file.

Creating a Tools Properties File

You can create a properties file with a text editor by specifying each argument, or option, using standard Java properties file format (name=value). For example, you can create a simple properties file that define a set of LDAP connection parameters as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
baseDN=dc=example,dc=com
```

Next, you can specify the location of the file using the --propertiesFilePath /path/to/ File option with the command-line tool. For example, if you save the previous properties file as `bin/mytool.properties`, you can specify the path to the properties file with `ldapsearch` as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/mytools.properties "(objectclass=*)"
```

Properties files do not allow quotation marks of any kind around values. Any spaces or special characters should be escaped. For example,

```
bindDN=cn=QA\ Managers,ou=groups,dc=example,dc=com
```

The following is not allowed as it contains quotation marks:

```
bindDN=cn="QA Managers,ou=groups,dc=example,dc=com"
```

Tool-Specific Properties

The Directory Proxy Server also supports properties for specific tool options using the format: `tool.option=value`. Tool-specific options have precedence over general options. For example, the following properties file uses `ldapsearch.port=2389` for `ldapsearch` requests by the client. All other tools that use the properties file uses `port=1389`.

```
hostname=server1.example.com
port=1389
ldapsearch.port=2389
bindDN=cn=Directory\ Manager
```

Another example using the `dsconfig` configuration tool is as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
dsconfig.bindPasswordFile=/ds/config/password
```

Note: The `.bindPasswordFile` property requires an absolute path. If you were to specify `~/ds/config/password`, where ~ refers to the home directory, the server does not expand the ~ value when read from the properties file.

Specifying Default Properties Files

The Directory Proxy Server provides a default properties files that apply to all command-line utilities used in client requests. A default properties file, `tools.properties`, is located in the `<server-root>/config` directory.

If you place a custom properties file that has a different filename as `tools.properties` in this default location, you need to specify the path using the --propertiesFilePath option. If you make changes to the `tools.properties` file, you do not need the --propertiesFilePath option. See the examples in the next section.

Evaluation Order Summary

The Directory Proxy Server uses the following evaluation ordering to determine options for a given command-line utility:

- All options used with a utility on the command line takes precedence over any options in any properties file.
- If the --propertiesFilePath option is used with no other options, the Directory Proxy Server takes its options from the specified properties file.

- If no options are used on the command line including the --propertiesFilePath option (and --noPropertiesFile), the Directory Proxy Server searches for the `tools.properties` file at `<server-root>`
- If no default properties file is found and a required option is missing, the tool generates an error.
- Tool-specific properties (for example, `ldapsearch.port=3389`) have precedence over general properties (for example, `port=1389`).

Evaluation Order Example

Given the following properties file that is saved as `<server-root>/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

The Directory Proxy Server locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (port and baseDN). The command uses the `bindDN` and `bindPassword` arguments specified in the properties file.

   ```
   $ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
     --propertiesFilePath bin/tools.properties “(objectclass=*)”
   ```
2. Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the Directory Proxy Server uses the specified properties file as follows:

   ```
   $ bin/ldapsearch --propertiesFilePath bin/tools.properties \
     “(objectclass=*)”
   ```
3. If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, the Directory Proxy Server attempts to locate any default `tools.properties` file in the following location:

   ```
   <server-root>/config/tools.properties
   ```

   Assume that you move your tools.properties file from `<server-root>/bin` to the `<server-root>/config` directory. You can then run your tools as follows:

   ```
   $ bin/ldapsearch "(objectclass=*)"
   ```

   The Directory Proxy Server can be configured so that it does not search for a properties file by using the --noPropertiesFile option. This options tells the Directory Proxy Server to use only those options specified on the command line. The --propertiesFilePath and --noPropertiesFile options are mutually exclusive and cannot be used together.
4. If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Running Task-based Utilities

The Directory Proxy Server has a Tasks subsystem that allows you to schedule basic operations, such as backup, restore, bin/start-server, bin/start-server and others. All task-based utilities require the --task option that explicitly indicates the utility is intended to run as a task rather than in offline mode. The following table shows the arguments that can be used for task-based operations:

Table 14: Task-based Utilities

Option	Description
--task	Indicates that the tool is invoked as a task. The --task argument is required. If a tool is invoked as a task without this --task argument, then a warning message will be displayed stating that it must be used. If the --task argument is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message will be displayed and the tool will exit with an error.
--start	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation starts when scheduled as a server task. A value of '0' causes the task to be scheduled for immediate execution. When this option is used, the operation is scheduled to start at the specified time, after which this utility will exit immediately.
--dependency	Specifies the ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution. This option can be used multiple times in a single command.
--failedDependencyAction	Specifies the action this task will take should one of its dependent tasks fail. The value must be one of the following: `PROCESS`, `CANCEL`, `DISABLE`. If not specified, the default value is `CANCEL`. This option can be used multiple times in a single command.
--completionNotify	Specifies the email address of a recipient to be notified when the task completes. This option can be used multiple times in a single command.
--errorNotify	Specifies the email address of a recipient to be notified if an error occurs when this task executes. This option can be used multiple times in a single command.

PingDirectory™

Release 7.0.1

Consent Solution Guide

Ping Identity®

Notice

PingDirectory™ Product Documentation

Trademarks

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Contents

Chapter

1

Introduction to the Consent Service and Consent API

Topics:

Companies gain loyalty and trust when they offer transparency and control to their users and customers regarding the personal data that is collected, processed, or shared. In Europe, the General Data Protection Regulation (GDPR) was designed specifically for allowing companies to collect and use valuable data about its users, while protecting the rights of citizens to control what is collected and used. To support the collection and end-user control of personal data, PingDirectory Server includes schema and REST APIs that provide the ability to collect fine-grained data authorizations (consents), from users and customers.

Consent Service overview

The Consent Service is an HTTP-based REST API hosted by the PingDirectory Server or PingDirectoryProxy Server. The service enables the collection of consent from application users, the enforcement of consent, a user's management of his or her consent, and auditing of consent actions. Enterprises can integrate these features into their applications to give users transparency and control of their data privacy.

For the purpose of this document, the following terms are used:

Table 1: Consent Terms

Term	Meaning
Consent definition	The terms of the fine-grained contract, which describes the data that can be processed or shared, and a purpose for processing or sharing the data. The consent definition is stored in the server configuration.
Consent localization	A child object of a definition that contains versioned, localized text for the consent definition, to be used when prompting an individual. This is stored in the server configuration.
Consent record	A record of a consent interaction with a user. Consent records are stored in the directory tree.
Subject	The individual whose data can be collected, processed, or shared.
Actor	The individual who granted/denied/revoked consent. This is usually the same as the subject.
Audience	The entity, application, or service that is granted or denied access to a subject's data for a specific purpose.

Consent API overview

The PingDirectory Server and PingDirectoryProxy Server provide a REST API for managing individuals' consent to handle their data. This can be used as a component of a larger solution, such as a GDPR compliance system or the PingDataGovernance Server Open Banking Account Requests API.

The PingDirectory Server Consent API enables authorization services:

- to capture user consent for sharing or processing data
- to confirm that consent to share or process data has been granted
- for individuals to manage the consent that they have granted.

Detailed API documentation can be found on the Ping Identity website.

How consents are collected

User consent is collected by creating a consent record through the Consent API. In most cases, the Consent API client uses consent localization data to construct an approval prompt to display to the user. This prompt should include text describing what data is collected and for what purpose, allowing the user to make an informed decision about the value of sharing his or her data.

For example, a web application needing to collect consent for a user's browsing behavior would use the Consent API to look up the localizations for the `browsing-behavior` consent definition. It would select the localization appropriate for the user and use data from the localization resource to construct a consent prompt for display to the user. After the user is prompted and makes a decision, the client could store the decision by creating a new consent record through the Consent API.

How consents are enforced

The Consent Service can be used as a data source for making access control decisions. If a particular data usage scenario requires consent, then the application or service needing to access or process that data must not be able to use the data unless the user has provided consent. The entity that performs this consent check may be the application itself or some other service.

To perform a consent check, the Consent API client must be able to correlate a data access request type with a consent definition. For example, if a web application needs to collect a user's browsing behavior, this data collection scenario might be represented by a consent definition called `browsing-behavior`. The application would check for an existing consent grant by searching the Consent API for a consent record that matches the user and the `browsing-behavior` consent definition. If a match is found, then the application can proceed. If a match is not found, the application must collect consent from the user.

How applications use the Consent API

The following example illustrates both consent capture and consent enforcement. This example follows a user's journey on a website during which the company must gather consent to track the user's browsing behavior:

1. A user launches the company's application and authenticates. The application wants to record the page visit, but first it must check if the user has granted consent to do so.
2. The application makes a call to the Consent API to determine if the `browsing-behavior` consent record exists for this user, and whether consent been granted.
3. The API returns a result indicating that no consent record exists. The application must prompt the user for his or her consent. The application calls the Consent API to retrieve the localization for the `browsing-behavior` consent, which includes the language that the application uses to produce a prompt for the user.
4. After the user makes a decision, the application stores the user's decision by creating a new consent record. This is through a call to the Consent API.
5. Later, the user visits another page in the company's site. The application wants to record the page visit, and again checks whether the user has granted consent to do so.
6. The application makes a call to the Consent API to get the `browsing-behavior` consent record for this user.
7. If the user's consent record agrees to have the company track his or her browsing behavior, the application can then make the appropriate calls to track browsing behavior. This is consent enforcement.

Chapter

2

Consent Service configuration

Topics:

This section provides details for installing and configuring the components on which the Consent Service relies. Refer to the PingDirectory Server Administration Guide for detailed configuration information.

Configuration overview

The Consent Service is not enabled by default. The setup and configuration process varies depending on the following factors:

- Whether client applications will allow an individual to self-manage consents.
- Whether some or all client applications will be privileged, with the ability to manage all consents.
- The HTTP authentication method used by client applications.
- Whether consent records exist in the same directory as user entries.

Example configuration scenarios

The following client application scenarios are available for determining how the Consent Service should be configured to meet your business needs.

Directly managed consents

In this scenario, one or more client applications provide provide an interface for individuals to directly manage their own consent records. These applications can only manage consents for the currently authenticated user. In addition, there is also a client application for consent administrators. An OAuth 2 authorization server grants access tokens that the applications uses to access the Consent API.

Configuration for this scenario includes:

1. Configure an OAuth 2 authorization server to issue a `urn:pingdirectory:consent` scope to individuals and a `urn:pingdirectory:consent_admin` scope to consent administrators.
2. Create an identity mapper to map subject identifiers used by the authorization server to LDAP DNs used by the PingDirectory Server.
3. Configre an access token validator to validate tokens issued by the OAuth 2 authorization server.
4. Configure the Consent HTTP Servlet Extension to disable HTTP basic authentication and restart the HTTPS Connection Handler.
5. Configure the Consent Service to use the OAuth scopes, token validator, and identity mapper.

Indirectly managed consents (basic authentication)

In this scenario, an application uses a privileged service account to manage its users' consents. The application's privileged account can access any consent record, which gives the application the ability to perform operations that an individual user cannot. The following include steps the setup needed for the PingDataGovernance Server's Open Banking Account Requests service to use the Consent Service as its backend.

Configuration for this scenario includes:

1. Create a service account for the application.
2. Configure the Consent HTTP Servlet Extension to enable HTTP basic authentication and restart the HTTPS Connection Handler.
3. Create an identity mapper to map consent record subject and actor attribute values to LDAP DNs. This is optional.
4. Configure the Consent Service to use the application's service account, and optionally the identity mapper.

Set up with the configuration scripts

PingDirectory Server includes two configuration scripts that can serve as the starting point for setting up the Consent Service. Both scripts must be carefully reviewed and updated to support your client application scenarios and business needs.

- `consent-service-base-entries.ldif` - This LDIF script can be imported to create the base DN where consent records will be stored.
- `consent-service-cfg.dsconfig` - This script can be imported to configure and enable the Consent Service.

Both are located in the `/resource/consent/` directory of the PingDirectory Server server root.

Basic configuration with the `consent-service-base-entries.ldif` file includes:

1. Edit the LDIF script and change the location of where consent records will be stored.
2. Import the LDIF script using the `ldapmodify`command, such as:

```
$ bin/ldapmodify --defaultAdd \
  --filename consent-service-base-entries.ldif
```

Basic configuration with the `consent-service-cfg.dsconfig` file includes:

1. Search for `CHANGE-ME` and replace values.
2. Review configuration commands and make additional changes to match existing Ping environment parameters, application scenarios, and business needs.
3. Impost the script with the `dsconfig` command, such as:

```
$ bin/dsconfig --no-prompt \
  --batch-file consent-service-cfg.dsconfig
```

Setup in a replicated PingDirectory Server environment

Running the Consent Service setup script requires special consideration in an environment that includes replicated PingDirectory Servers. If possible, setup the Consent Service after replication is enabled for the PingDirectory Servers. See the *PingDirectory Server Administration Guide* for details about server replication.

Set up Consent Service after replication is enabled

Complete the following steps if replication is already enabled for PingDirectory Servers.

1. If needed, configure the PingDirectory Servers to use a configuration group called "all-servers." This will ensure that configuration changes are applied to all servers in a topology.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:all-servers
```

2. Run the Consent Service setup script.

```
$ bin/dsconfig --no-prompt \
  --batch-file resource/consent/consent-service-cfg.dsconfig
```

Set up Consent Service before replication is enabled

If you have already set up the Consent Service on a standalone PingDirectory Server, perform the following the steps before enabling replication. In this example, "DS1" is the original PingDirectory Server, and "DS2" is the second server that will be added as a replica.

1. Run the `config-diff` command without arguments on DS1 to produce a batch file that contains configuration changes that will be applied to DS2.

```
$ bin/config-diff > config-changes.dsconfig
```

2. Apply the `config-changes.dsconfig` file to DS2.

```
$ bin/dsconfig --no-prompt \
  --batch-file config-changes.dsconfig \
  --applyChangeTo single-server
```

3. Restart DS2.
4. Enable replication between the two servers.

Configuration reference

There are many configuration options for the Consent Sevice and application integration. The configuration scripts included with the PingDirectory Server provide a starting point. Additional detailed information about the Consent Service properties and configuration is provided as reference.

General Consent Service configuration

The Consent Service configuration is used to control authorization behavior and determines where consent records are stored in the PingDirectory Server. The service properties are configured with the `dsconfig set-consent-service-prop` command. The consent service configuration script configures the consent service properties as follows:

```
$ bin/dsconfig set-consent-service-prop \
  --set enabled:true \
  --set base-dn:ou=consents,dc=example,dc=com \
  --set "bind-dn:cn=consent service account" \
  --set unprivileged-consent-scope:urn:pingdirectory:consent \
  --set privileged-consent-scope:urn:pingdirectory:consent_admin \
  --set "consent-record-identity-mapper:User ID Identity Mapper"
```

The following are Consent Service properties.

Table 2: Consent Service properties

Property	Description	Required to enable service
`enabled`	If set to true, enables the Consent Service for handling client requests.	Yes
`base-dn`	Specifies a container DN for consent record entries.	Yes
`bind-dn`	Specifies an internal service account used by the Consent Service to perform LDAP operations.	Yes
`service-account-dn`	Specifies one or more DNs of requesters that will be considered privileged when using basic authentication. If not defined, a requester will only be considered privileged if it is mapped to a DN with the `bypass-acl` privilege. Optional.	No
`unprivileged-consent-scope`	Specifies the name of the scope required for bearer tokens representing unprivileged requesters.	Yes
`privileged-consent-scope`	Specifies the name of the scope required for bearer tokens representing privileged requesters.	Yes
`consent-record-identity-mapper`	Specifies one or more identity mappers used to map consent record `subject` and `actor` values to DNs. If not defined,	No

Property	Description	Required to enable service
	the Consent Service will not handle requests from unprivileged clients. Optional.	
`audience`	Specifies an `audience` claim value that the Consent Service will require to be present in bearer tokens that it accepts. Optional.	No

For the Consent Service to report itself as available to clients, the following must be true:

- The Consent Service must be enabled.
- The Consent Service base DN must be configured and must exist.
- The internal service account must be configured and must exist.
- The internal service account must have the right to read, add, modify, and delete entries under the Consent Service base DN.

Create a container entry for consent records

Each consent record is a distinct entry in the PingDirectory Server, and the Consent Service requires that these entries be stored under a common base DN, defined by the `base-dn` property of the Consent Service configuration. The Consent Service LDIF file sets the base DN. Use these steps to choose a different location to store consent records.

1. To create the Consent Service base DN, open a text editor and save the following to the file `consent-service-base-dn.ldif`.

```
dn: ou=consents,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: consents
```

2. Use `ldapmodify` to add the entry.

```
$ bin/ldapmodify --defaultAdd --filename consent-service-base-dn.ldif
```

Create an internal service account

The Consent Service uses an internal LDAP connection to operate against consent records that are stored as LDAP entries. It authenticates this LDAP connection using a service account, which must be created and dedicated solely to the Consent Service.

The Consent Service configuration script configures the internal service account using a topology admin user. If needed, this can be changed to a root DN user or a user DN whose entry is in the user backend. In all cases, the service account should exist in every LDAP server in the topology.

This service account must have full read and write access to the Consent Service base DN, the ability to read users' `isMemberOf` attribute, and the right to use the following LDAP controls:

- IntermediateClientRequestControl (1.3.6.1.4.1.30221.2.5.2)
- NameWithEntryUUIDRequestControl (1.3.6.1.4.1.30221.2.5.44)
- RejectUnindexedSearchRequestControl (1.3.6.1.4.1.30221.2.5.54)
- PermissiveModifyRequestControl (1.2.840.113556.1.4.1413)
- PostReadRequestControl (1.3.6.1.1.13.2)

For more information about configuring access, see the *"Managing Access Contro"l* chapter of the PingDirectory Server Administration Guide.

- To ensure the correct access, create a user with the `bypass-acl` privilege. The following `dsconfig` command creates a topology admin user with the `bypass-acl` privilege. After this is created, set this user as the `bind-dn` for the Consent Service.

```
$ dsconfig create-topology-admin-user \
  --user-name "Consent Service Account" \
  --set "description:Consent API service account" \
  --set "alternate-bind-dn:cn=consent service account" \
  --set first-name:Consent \
  --set inherit-default-root-privileges:false \
  --set last-name:Service \
  --set password:CHANGE-ME \
  --set privilege:bypass-acl
```

- Because the `bypass-acl` privilege grants a broad level of access, you may not want to grant this privilege to the Consent Service account. If desired, add the following ACI to enable a targetted set of functionality for the Consent Service. The following example grants this access to the DN `cn=consent service account` using global ACIs:

```
# Grant access to the consent record base DN ou=consents,dc=example,dc=com
dsconfig set-access-control-handler-prop --add 'global-aci:(target="ldap:///
ou=consents,dc=example,dc=com")(targetattr="*||+")(version 3.0; acl "Consent
 Service account access to consent record data"; allow(all) userdn="ldap:///
cn=consent service account";)'

# Grant access to the LDAP request controls used by the Consent Service.
dsconfig set-access-control-handler-prop --add 'global-aci:
(targetcontrol="1.3.6.1.4.1.30221.2.5.2||1.3.6.1.4.1.30221.2.5.44||
1.3.6.1.4.1.30221.2.5.54||1.2.840.113556.1.4.1413||1.3.6.1.1.13.2")(version
 3.0; acl "Consent Service account access to selected controls"; allow
 (read) userdn="ldap:///cn=consent service account";)'

# Grant access to use the Effective Rights Control.
dsconfig set-access-control-handler-prop --add 'global-aci:
(targetcontrol="1.3.6.1.4.1.42.2.27.9.5.2")(version 3.0; acl "Consent
 Service account access to Get Effective Rights control"; allow (read)
 userdn="ldap:///cn=consent service account";)'
```

Configure an identity mapper

The Consent Service uses identity mappers to map requester identities, subject values, and actor values to DNs. An identity mapper takes a user identifier string and correlates the identifier with the DN of a user entry. The PingDirectory Server provides four different types of identity mappers.

Table 3: Identity mappers

Identity mapper type	Description
Exact match identity mapper	Maps a user identifier to a DN by searching for an entry with an attribute that exactly matches the identifier.
Regular expression identity mapper	Similar to an exact match identity mapper, but allows a regular expression to be specified for more flexible matching.
Third-party identity mapper	A custom Java identity mapper implementation written using the Server SDK.
Groovy scripted identity mapper	A custom Groovy identity mapper implementation written using the Server SDK.

The Consent Service can be configured to use identity mappers for each of the following scenarios:

- Requesters authenticating using basic authentication - use the Consent HTTP Servlet Extension `identity-mapper` property to configure an identity mapper that takes the HTTP Basic authorization username string to find the corresponding user's identity in the PingDirectory Server.

- Requesters authenticating using bearer token authentication - use the Access Token Validator `identity-mapper` property to configure an identity mapper that takes the subject (or other claim value from the OAuth token) to find the corresponding user's identity in the PingDirectory Server.
- Consent record actor and subject values - use the Consent Service `consent-record-identity-mapper` property to configure an identity mapper that takes these consent record attribute values and uses them to find the corresponding users' identities in the PingDirectory Server.

The Consent Service configuration script configures a single identity mapper to be used for all three scenarios. The provided identity mapper searches by `uid`, `cn`, or `entryUUID` attributes under the base DNs `cn=config and ou=people,dc=example,dc=com`.

The following configuration provides an example of an identity mapper that will match a user identifier to an LDAP entry with the same value in its `uid` attribute:

```
$ bin/dsconfig create-identity-mapper --mapper-name "User ID Exact Match" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:uid
```

The following configuration shows another typical example, that of an identity mapper that will match a user identifier to an LDAP entry with the same value in its `entryUUID` attribute:

```
$ bin/dsconfig create-identity-mapper --mapper-name "EntryUUID Exact Match"
 \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:entryUUID
```

The last example creates an identity mapper that will match a user identifier to an LDAP entry with the same value in either its `uid`, `cn`, or `entryUUID` attribute. This identity mapper will also constrain its search to the `ou=people,dc=example,dc=com` and `cn=config` base DNs. (The `cn=config` base DN is not searched by default, and must be explicitly listed to be searched.)

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "User ID Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:uid \
  --set match-attribute:cn \
  --set match-attribute:entryUUID \
  --set match-base-dn:cn=config \
  --set match-base-dn:ou=people,dc=example,dc=com
```

Authentication methods

The Consent Service supports two HTTP authentication methods, which are both enabled by default:

- Basic authentication
- Bearer token authentication

The Consent servlet looks at the request's Authorization header to determine which authentication type is being used by the client.

With basic authentication, the client provides an encoded username/password pair in the HTTP Authorization request header. When the Consent Service receives a request using basic authentication, it maps the username credential to a DN using an identity mapper. This DN is designated the `auth DN` and is used to make subsequent authorization decisions. The Consent Service then performs an LDAP bind using the DN and password to determine if the request can be processed.

With bearer token authentication, the client provides an access token in the HTTP Authorization request header. The access token is always obtained by the client from an external OAuth 2 authorization server and encapsulates information ("claims") about a user identity, the client identity, and the requests that the client is authorized to make.

The PingDirectory Server must be configured to accept access tokens using one or both available access token validators:

- **PingFederate access token validator**. Supports access tokens issued by a PingFederate authorization server. This validator verifies an access token and discovers its claims by making a request to the PingFederate server's token introspection endpoint.
- **JWT access token validator**. Supports signed or encrypted JWT access tokens issued by an arbitrary authorization server. This validator checks an access token by cryptographically verifying the token's signature using a trusted public certificate. The token's claims are encoded in the token itself, so discovering the token's claims does not require an outgoing token introspection request.

The token validator uses its identity mapper to map the subject claim to a DN. This DN is designated the `auth DN` and is used along with the token's claims to make subsequent authorization decisions.

If the PingDirectory Server is configured with at least one access token validator, it will be used by the Consent Service. If the PingDirectory Server is configured with more than one access token validator, the validators are consulted in order until one is able to successfully authenticate the request.

If the PingDirectory Server is configured with multiple access token validators, but only one should be used by the Consent Service, the access token validator can be configured by setting the `access-token-validator` property of the Consent HTTP Servlet Extension.

Note: Configuring an access token validator for the Consent Service requires information from the authorization server configuration:

- The values that the authorization server sets for `subject` claims must be mappable to a DN in the PingDirectory Server.
- The authorization server must be configured to authorize clients and grant scopes appropriately for privileged or unprivileged Consent API access.
- The authorization server must be configured to issue tokens with scopes corresponding to the Consent Service's `unprivileged-scope-name` and `privileged-scope-name` configuration.

Refer to the authorization server's documentation for guidance.

Configure basic authentication

Basic authentication is enabled by default, and the settings are configured in the Consent HTTP Servlet Extension configuration.

- Use the following command to disable basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set basic-auth-enabled:false
```

- Use the following command to enable basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set basic-auth-enabled:true
```

- Use the following command to configure an identity mapper for basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "identity-mapper:User ID Exact Match"
```

- All of these configuration changes require the Consent servlet to be reloaded before they can take effect. Use the following commands to restart the connection handler that hosts the Consent servlet.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:false
```

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true
```

Configure bearer token authentication

- The following is an example access token validator configured on the PingDirectory Server for a PingFederate server:

```
$ bin/dsconfig create-external-server \
  --server-name PingFederate \
  --type http \
  --set base-url:https://my-ping-federate-server:1443/
```

```
$ bin/dsconfig create-access-token-validator \
  --validator-name "PingFederate Token Validator" \
  --type ping-federate \
  --set enabled:true \
  --set "identity-mapper:User ID Exact Match" \
  --set authorization-server:PingFederate \
  --set client-id:id \
  --set client-secret:secret
```

- If more than one access token validator is configured on the PingDirectory Server, the Consent Service can be configured to use a single validator with the following command:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "access-token-validator:PingFederate Token Validator"
```

Configure Consent Service scopes

The Consent Service checks access tokens for a `subject` claim and uses an identity mapper to map the value to a DN, called the request DN or auth DN. If no request DN can be mapped, the request is rejected. In addition, the Consent Service will only accept an access token with a scope that it is configured to recognize.

- An unprivileged consent scope designates the requester as unprivileged. The scope's name is configured with the Consent Service's `unprivileged-consent-scope` property.
- A privileged consent scope designates the requester as privileged. This is configured using the Consent Service's `privileged-consent-scope` property.

The authorization server must also be configured to issue tokens with these scopes.

- The following example configures these scopes for the Consent Service.

```
$ bin/dsconfig set-consent-service-prop \
  --set unprivileged-consent-scope:consent \
  --set privileged-consent-scope:consent_admin
```

Authorization

The Consent Service's distinction between privileged and unprivileged requesters determines the type of operations that can be performed by requesters. During the authorization phase, the Consent servlet performs checks on both the bearer token claims (if present) and the `auth DN` to determine if the requester is privileged or unprivileged. These are summarized in the following table.

Table 4: Available operations per requester type

Requester type	Description	Access determined by	Can create consent records	Can update consent records	Can delete consent records
Unprivileged	Requesters with no authority to operate on consent records other than their own. Unprivileged requesters can only be used if the Consent Service is configured to map subject and actor DNs using the `consent-record-identity-mapper` property.	A requester is considered unprivileged if it does not meet any of the criteria for a privileged requester. If using bearer token authentication, the access token must include a scope named by the `unprivileged-consent-scope` property of the Consent Service configuration. Also, an unprivileged requester can only perform actions on consent records where the subject DN and actor DN match the requester DN.	Yes, if the requester DN matches the subject DN and actor DN.	Yes, if the requester DN matches the subject DN and actor DN.	No.
Privileged	A requester with the authority to perform any operation on any consent record.	When using basic authentication, a requester is considered privileged if the requester DN either has the `bypass-acl` privilege or is listed in the `service-account-dn` property of the Consent Service configuration. If using bearer token authentication, the access token must include a scope named by the `privileged-consent-scope` property of the Consent Service configuration.	Yes.	Yes.	Yes.

Bearer token check

If a bearer token was used, the following checks are performed:

- If the Consent Service's `audience` property is configured, the bearer token's audience claim must match the configured value.
- If the bearer token contains a scope matching the Consent Service's `privileged-scope-name` property, then the requester is considered privileged.
- If not, the bearer token must have a scope matching the Consent Service's `unprivileged-scope-name` property, and the requester is considered unprivileged.

Basic authentication check

If basic authentication is used, the following checks are performed:

- If the `auth DN` has the LDAP privilege `bypass-acl`, the requester is privileged.
- If the `auth DN` is listed in the Consent Service's `service-account-dn` property, the requester is privileged.
- If not, the requester is considered unprivileged.

Chapter 3

Manage consents

Topics:

This section describes the tasks required to support the collection and end-user control of personal data, and manage users' consents.

Overview of consent management

The full lifecycle of consent management goes beyond collecting the user's consent. First, the terms of each consent contract must be centrally managed. After collecting consent, the user will want to review previously granted consents and potentially revoke some. Finally, companies will need to be able to trace the history of updates to any consent in order to resolve a dispute or respond to audit.

Consent definitions and localizations

Companies will want to centrally manage the language used when prompting a user to give consent. This is key to ensuring a consistent user experience across multiple applications, such as mobile and web. The Consent Service requires one or more consent definitions to be defined in the PingDirectory Server configuration. Each consent definition represents the combination of:

- The data to be collected or shared.
- The purpose for collecting or sharing this data.

For example, a consent definition could represent user email addresses, used to deliver a third party's email newsletter. A consent definition could also represent access to a user's network-connected IoT device, which would be used for a home automation task controlled by a third party.

Each consent definition must have one or more localization. A localization is a versioned object consisting of the data that a Consent API client needs to prompt a user for consent. When a consent record is accepted or denied by a Consent Service client, it must include a reference to a consent definition, locale, and version.

Create consent definition and localization

- The following creates a consent definition and a localization for it.

```
$ bin/dsconfig create-consent-definition \
  --definition-name email_newsletter \
  --set "display-name:Email newsletter"
```

```
$ bin/dsconfig create-consent-definition-localization \
  --definition-name email_newsletter \
  --localization-name en-US \
  --set version:1.0 \
  --set "data-text:Your email address" \
  --set "purpose-text:To receive newsletter updates"
```

- The following example updates a localization and its version.

```
$ bin/dsconfig set-consent-definition-localization-prop \
  --definition-name email_newsletter \
  --localization-name en-US \
  --set version:1.1 \
  --set "data-text:Your preferred email address"
```

Perform an audit on consents

Changes to Consent Service resources are tracked by one of two types of audit logs. For examples of configuring either type of log, see the `<server-root>/resource/consent-service-cfg.dsconfig` script bundled with the server or *Logging*. This example uses the Consent Trace Logger. It represents Consent Service change events using the same field names used by the Consent API.

Table 5: Log Publishers

Log publisher	Log publisher type	Description
Consent Trace Logger	file-based-trace	Records Consent Service events at the Consent API level. Change events are recorded using messages of type `audit`.
Consent LDAP Audit Logger	file-based-audit	Records data changes at the LDAP level. In combination with a Request Criteria configuration object, an LDAP audit logger can be configured to record changes to Consent Service resources only.

Trace logger keys for auditing

Trace logger audit messages consist of a timestamp, the message type (`CONSENT AUDIT`), and a set of key/value pairs. A subset of important keys are described in the following table.

Note: The keys used in trace log audit messages vary depending on the type of resource.

Table 6: Log Publishers

Trace logger key	Description
`requestID`	A server-specific HTTP request ID. This value can be correlated with messages produced by other loggers.
`resourceType`	The type of Consent Service resource that was changed. Possible values are `definition`, `localization`, or `consent`.
`changeType`	The type of change recorded by this message. Possible values are `create`, `update`, or `delete`.
`attrsAdded`	A comma-delimited list of the attributes that were added to the resource.
`attrsUpdated`	A comma-delimited list of the attributes that were modified on the resource.
`attrsDeleted`	A comma-delimited list of the attributes that were removed from the resource.
`requestDN`	The DN of the requester, which is available only when the resource type is `consent`.
`definitionID`	The consent definition ID. If the resource type is `definition`, this identifies the definition that was changed. If the resource type is `localization`, this identifies the parent definition. If the resource type is `consent`, this identifies the consent record's related definition.
`locale`	The locale. If the resource type is `localization`, this identifies the localization (in combination with the definition ID). If the resource type is `consent`, this identifies the related localization (combined with the definition ID).
`consentID`	The consent record ID, available only when the resource type is `consent`.
`subject`	The subject value, available only when the resource type is `consent`.
`subjectDN`	The subject's mapped LDAP DN, available only when the resource type is `consent`.
`actor`	The actor value, available only when the resource type is `consent`.
`actorDN`	The actor's mapped LDAP DN, available only when the resource type is `consent`.

Trace logger key	Description
audience	The audience value, available only when the resource type is consent.
status	The consent status. Possible values are pending, accepted, denied, revoked, and restricted. Only available when the resource type is consent.
previousStatus	The previous consent status, if applicable. Only available when the resource type is consent.
msg	A multiline value that includes the complete body of the changed resource. If the action is an update or a delete, the resource's body before the change will be included.

Perform an audit

Consent resource changes for particular entities (such as a specific user, or a specific consent definition) can be audited by searching the trace log using a combination of one of the message keys and the desired value. For example, if an individual's LDAP DN is known, then the subjectDN key can be used to construct a text search for any audit log messages containing that DN. Any matching log messages would constitute a history of that individual's consent activity.

Example new consent record

The following is a sample record. this audit log message provides important values in a parseable key/value format, but also includes the entirety of the new consent record.

```
[22/May/2018:18:02:42.584 -0500] CONSENT AUDIT requestID=57
 requestDN="uid=user.0,ou=people,
  dc=example,dc=com" consentID="6cff325b-e092-4094-b7f9-5a30864b0d24"
 subject="user.0" subjectDN="uid=user.0,
  ou=People,dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com" audience="client1"
  definitionID="cats" locale="en-US" status="accepted"
 attrsAdded="actor,audience,createdDate,dataText,subject,
  purposeText,definition,id,updatedDate,actorDN,status,subjectDN"
 changeType="create" resourceType="consent" msg="
New Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'accepted','subject':'user.0','subjectDN':'uid=user.0,

 ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=People,dc=exampl
  'client1','definition':{'id':'cats','version':'1.0','locale':'en-
US'},'dataText':'Collect data about your
  cats','purposeText':'To recommend cat food flavors that will satisfy and
 delight your feline companion',

 'createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:02:42.553Z'}"
```

Example updated consent record

This example shows the complete consent record before and after it was updated. With the attrsUpdated, status, and previousStatus keys, one can determine that the status changed from accepted to revoked.

```
[22/May/2018:18:05:08.660 -0500] CONSENT AUDIT requestID=59
 requestDN="uid=user.0,ou=people,
   dc=example,dc=com" consentID="6cff325b-e092-4094-b7f9-5a30864b0d24"
 subject="user.0" subjectDN="uid=user.0,
   ou=People,dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com"
```

```
   audience="client1" definitionID="cats" locale="en-US" status="revoked"
 previousStatus="accepted"
   attrsUpdated="status" changeType="update" resourceType="consent" msg="
Previous Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'accepted','subject':'user.0','subjectDN':'uid=user.0,

 ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=People,dc=exampl
  'audience':'client1','definition':
{'id':'cats','version':'1.0','locale':'en-US'},'dataText':'Collect
  data about your cats','purposeText':'To recommend cat food flavors that
 will satisfy and delight your
  feline
 companion','createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:02:42.
Updated Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'revoked','subject':'user.0','subjectDN':

 'uid=user.0,ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=Peop
  dc=com','audience':'client1','definition':
{'id':'cats','version':'1.0','locale':'en-US'},'dataText':
  'Collect data about your cats','purposeText':'To recommend cat food
 flavors that will satisfy and
  delight your feline
 companion','createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:05:08.
```

Example deleted consent record

This example shows that a consent record has been deleted, and the complete representation of the consent record prior to its deletion is provided.

```
[22/May/2018:18:06:35.071 -0500] CONSENT AUDIT requestID=61
 requestDN="cn=directory manager"
   consentID="6cff325b-e092-4094-b7f9-5a30864b0d24" subject="user.0"
 subjectDN="uid=user.0,ou=People,
   dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com" audience="client1"
   definitionID="cats" locale="en-US" status="revoked"
 previousStatus="revoked" attrsDeleted="actor,audience,

 createdDate,dataText,subject,purposeText,definition,id,updatedDate,actorDN,status,subje
 changeType="delete"
   resourceType="consent" msg="
Deleted Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'revoked','subject':'user.0','subjectDN':

 'uid=user.0,ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=Peop
   dc=example,dc=com','audience':'client1','definition':
{'id':'cats','version':'1.0','currentVersion':
   '1.0','locale':'en-US'},'dataText':'Collect data about your
 cats','purposeText':'To recommend cat food
   flavors that will satisfy and delight your feline
 companion','createdDate':'2018-05-22T23:02:42.553Z',
   'updatedDate':'2018-05-22T23:05:08.655Z'}"
```

Logging

The PingDirectory Server trace log publisher is used for logging events generated by HTTP service operations. The trace logger can be used to observe, debug, and audit consent requests.

Note: To create a log of consent audit events only, remove all message types except for `consent-message-type:audit`.

- The following example of creates a trace logger for all consent events, plus summaries of HTTP requests and responses.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "Consent Trace Logger" \
  --type file-based-trace \
  --set "description:Records Consent API operations" \
  --set enabled:true \
  --set consent-message-type:audit \
  --set consent-message-type:consent-created \
  --set consent-message-type:consent-deleted \
  --set consent-message-type:consent-retrieved \
  --set consent-message-type:consent-search \
  --set consent-message-type:consent-updated \
  --set consent-message-type:definition-created \
  --set consent-message-type:definition-deleted \
  --set consent-message-type:definition-retrieved \
  --set consent-message-type:definition-search \
  --set consent-message-type:definition-updated \
  --set consent-message-type:error \
  --set consent-message-type:localization-created \
  --set consent-message-type:localization-deleted \
  --set consent-message-type:localization-retrieved \
  --set consent-message-type:localization-search \
  --set consent-message-type:localization-updated \
  --set http-message-type:request \
  --set http-message-type:response \
  --set 'exclude-path-pattern:/**/*.css' \
  --set 'exclude-path-pattern:/**/*.eot' \
  --set 'exclude-path-pattern:/**/*.gif' \
  --set 'exclude-path-pattern:/**/*.ico' \
  --set 'exclude-path-pattern:/**/*.jpg' \
  --set 'exclude-path-pattern:/**/*.js' \
  --set 'exclude-path-pattern:/**/*.png' \
  --set 'exclude-path-pattern:/**/*.svg' \
  --set 'exclude-path-pattern:/**/*.ttf' \
  --set 'exclude-path-pattern:/**/*.woff' \
  --set 'exclude-path-pattern:/**/*.woff2' \
  --set 'exclude-path-pattern:/console/**' \
  --set 'exclude-path-pattern:/console/**/template/**' \
  --set log-file:logs/consent-trace \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy"
```

Correlating user and consent data

In some cases, the organization that has been granted consent by a group of users may need to perform an LDAP search so that they can act upon consent data in the aggregate. For example, a marketing group has collected consent to send a newsletter by email. A search must be performed that will list all of the consent records where the consent definition is `email` and the status is `accepted`. Those records must be correlated to user entries, and each user's email address must be retrieved.

This task is performed with an LDAP search on the PingDirectory Server. Every consent record has a `subject`, the user whose data is collected and stored. The Consent Service can be configured so that it stores the subject's DN in the `subjectDN` field.

In the LDAP schema:

- A consent record's `subjectDN` field is the `ping-consent-subject-dn` attribute.
- A consent record's status is the `ping-consent-state` attribute.
- A consent record's definition ID is in the `ping-consent-definition.id` JSON attribute field.
- And a user entry's email address is in the `mail` attribute.

The search will need to find all of the consent record entries where `ping-consent-definition.id` is `email` and the `ping-consent-status` is `accepted`. It then needs to correlate those consent record entries to user entries using `ping-consent-subject-dn`, and retrieve each user entry's `mail` attribute value. For example:

```
$ bin/ldapsearch \
   --baseDN "ou=consents,dc=example,dc=com" \
   --searchScope sub \
   --joinRule "dn:ping-consent-subject-dn" \
   --joinBaseDN "ou=people,dc=example,dc=com" \
   --joinScope sub \
   --joinRequestedAttribute mail
   '&(ping-consent-
definition:jsonObjectFilterExtensibleMatch:={ "filterType" : "equals",
 "field" : "id", "value" : "email" })(ping-consent-state=accepted)' \
   1.1

   # Join Result Control:
   #      OID:  1.3.6.1.4.1.30221.2.5.9
   #      Join Result Code:  0 (success)
   #      Joined With Entry:
   #           dn: uid=user.0,ou=People,dc=example,dc=com
   #           mail: user.0@example.com
   dn: entryUUID=9e481010-8330-425a-
bbf1-6637de053d48,ou=Consents,dc=example,dc=com

   # Result Code:  0 (success)
   # Number of Entries Returned:  1
```

The output listed under "Join Result Control" specifies the mail value.

Troubleshooting

The following are general guidelines for troubleshooting the Consent Service and any connection issues. When evaluating the configuration, make sure these issues are addressed first:

- Is the Consent Service enabled?
- Does the Consent Service base DN exist?
- Does the Consent Service's service account have the correct permissions?
- If the Consent Service should accept bearer tokens:
  - Are one or more Access Token Validators correctly configured?
  - Are the identity mappers for the Access Token Validators configured correctly?
  - Are the authorization servers correctly configured to issue tokens that the Consent Service will accept? Check the `audience`, `privileged-consent-scope`, and `unprivileged-consent-scope` properties of the Consent Service configuration.
- If privileged users are defined, are the members of the LDAP group specified by the Consent Service configuration's `privileged-users-group-dn` property?
- If there are applications that allow individuals to manage their own consents, is the system properly configured to map `actor` and `subject` DNs? Check the Consent Service configuration's `consent-record-identity-mapper` property.

Error cases

Consent Service is unavailable

If the Consent Service is unavailable, check that the service is enabled and that the communcation with the service is available. Confirm that the service account for the Consent Service has been properly provisioned. If the Consent Service resides on a PingDirectoryProxy Server, make sure that the service account exists on the PingDirectoryProxy Server and all PingDirectory Server behind the PingDirectoryProxy Server.

Requester lacks sufficient rights to perform operation

A request may be rejected with a 403 for the following reasons:

- The bearer token does not contain a required scope. Check the `privileged-consent-scope` and `unprivileged-consent-scope` properties of the Consent Service configuration.
- The bearer token does not contain a required `audience` claim. Check the `audience` property of the Consent Service configuration.
- Authentication was successful, but the requester is unprivileged and attempted to perform an operation that only a privileged requester may perform. For example, it may have attempted to act upon a consent record that it does not own, or it may have attempted to delete a consent record.

When using basic authentication, the requester must be listed in the Consent Service configuration `service-account-dn` property to be considered privileged.

Subject and actor do not match

Only a privileged requester can create or modify a consent record whose `subject` and `actor` values do not match.

Unindexed search

The Consent Service will not allow a client to make an unindexed search. In most cases, a client should be able to fix this by refining the search. For example, if a search by `subject` would be unindexed, perform a search by `subject definition` ID.

Search size limit exceeded

The Consent Service caps the maximum number of records that can be returned in a search result using its `search-size-limit` configuration property. This limit can be increased, or the client may be able to refine the search to produce fewer results.

Index

PingDirectory™

Release 2.0

Delegated Admin Application Guide

Notice

PingDirectory™ Product Documentation

Trademarks

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Contents

Chapter 1

Introduction to the Delegated Admin application

Topics:

Simple account administration can be performed by a group of administrators through the Delegated Admin application. This application manages identity attributes stored in the Directory Server.

Delegated Admin overview

Many organizations need to responsibly delegate some of the responsibility of managing identities in the Directory Server to reduce some of the burden of recurring tasks, such as password resets or updating account data. These simple administrative tasks can be delegated to a subset of administrators.

The Directory Server can be installed with the Delegated Admin application, which enables delegated administration of identities for scenarios such as help desk or customer service representatives (CSR) initiating a password reset and unlock; an HR administrator updating an employee profile; or an application administrator updating identity attributes.

Prerequisites

The Delegated Admin application manages user identity data stored in one or more Directory Servers. After a Directory Server instance is installed, the Delegated Admin application can be installed. You will need the HTTPS port that was configured during Directory Server setup.

A PingFederate server is used as the identity provider for authentication and authorization. Follow the PingFederate product documentation for installation and configuration instructions. Additional information is provided here for configuration with the Directory Server and the Delegated Admin application.

Chapter

2

Delegated Admin installation and configuration

Topics:

The Delegated Admin application relies on a previously installed PingDirectory Server and PingFederate Server. This section includes the configuration needed to support the Delegated Admin application.

Configuration overview

The Delegated Admin application must have a PingDirectory Server, and PingFederate Server installed. See the documentation for each product for installation instructions.

Configuration to support the Delegated Admin application on the PingDirectory Server includes:

- Configure administrators as Delegated Admin users.
- Configure attributes and attribute searching.

Configuration on the PingFederate Server includes:

- Configure PingFederate as the identity provider for the Delegated Admin application.
- Configure PingFederate as the OAuth server for the Delegated Admin application.
- Register the Delegated Admin application as a client.
- Register the PingDirectory Server as an OAuth token validator.

Authentication

The delegated administrator logs in to the Delegated Admin application through the PingFederate Server, which is configured as the authentication server and OpenID Connect (OIDC) provider. PingFederate validates the user credentials against the PingDirectory Server, and then issues an access token. The application obtains an access token from the PingFederate server, which encapsulates information `"claims"` about the user identity. The Delegated Admin application then presents this token to the Directory Server in the HTTP Authorization request header.

Interaction with the PingDirectory Server

The PingDirectory Server is configured to accept access tokens using Access Token Validators. The values that the PingFederate Server sets for access token `subject` claims must be mappable to a DN in the PingDirectory Server. Setting up an access token validator for use with the Delegated Admin application requires some coordination with the server configuration. In the suggested default configuration, the access token contains the entryUUID of the administrator user entry in the `sub` field. This value is mapped back to a PingDirectory Server entry using an Exact Match Identity Mapper.

Authorization by PingDirectory Server

Once validated, the PingDirectory Server checks the Delegated Admin configuration for authorization of the delegated administrator. Users or groups of users are authorized as delegated administrators in the Directory Server Administration Console, or with the `dsconfig` tool.

Client configuration

The Delegated Admin application uses PingFederate to authenticate users within the PingDirectory Server. Before installing and configuring the Delegated Admin application, you must configure two OAuth clients within PingFederate. One client is the Delegated Admin application, which will obtain an OIDC token that describes the authenticated user. The second client is the PingDirectory Server itself, which will call PingFederate to validate the OIDC token passed from the Delegated Admin application.

Specifically, the Delegated Admin OAuth client must be configured so that:

- The client id is "dadmin" and it requires no client secret key.
- The redirect URL is `https://SERVER:PORT/delegator/*`, where `SERVER:PORT` are the public host and port to access the Delegated Admin installation.
- The grant type is Implicit.
- The OIDC policy uses JWT tokens, where the entryUUID of the user is passed through the `subject` claim of the OIDC token.

The PingDirectory Server OAuth client must be configured so that:

- The client ID is "pingdirectory," and it does require a secret key.
- The grant type is Access Token Validation.

A detailed example of how PingFederate can be configured is available at the end of this guide.

Setup in a replicated PingDirectory Server environment

Running the Delegated Admin setup script requires special consideration in an environment that includes replicated PingDirectory Servers. If possible, setup the application after replication is enabled for the PingDirectory Servers. See the *PingDirectory Server Administration Guide* for details about server replication.

Set up Delegated Admin in an existing replicated topology

Complete the following steps if replication is already enabled for PingDirectory Servers.

1. If needed, configure the PingDirectory Servers to use a configuration group called "all-servers." This will ensure that configuration changes are applied to all servers in a topology.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:all-servers
```

2. Run the Delegated Admin setup script. See *Install the Delegated Admin application*.

Add an additional server to a replicated topology

To add an additional server to a topology containing a server already configured with the Delegated Admin application, perform the following the steps. In this example, "DS1" is the original PingDirectory Server, and "DS2" is the second server that will be added as a replica.

1. Run the `config-diff` command without arguments on DS1 to produce a batch file that contains configuration changes that will be applied to DS2.

```
$ bin/config-diff > config-changes.dsconfig
```

2. Apply the `config-changes.dsconfig` file to DS2.

```
$ bin/dsconfig --no-prompt \
  --batch-file config-changes.dsconfig \
  --applyChangeTo single-server
```

3. Restart DS2.
4. Enable replication between the two servers.

Install the Delegated Admin application

After the PingDirectory Server and the PingFederate Server have been configured, install the Delegated Admin application.

For systems such as Microsoft Windows that do not support bash scripts, copy the `delegated-admin-template.dsconfig` file and replace variables (`${variable}`) with actual values.

Note: If installing the Delegated Admin application on a PingDirectory Server that had "install with sample data" chosen as an installation option, the following ACI must be removed from the PingDirectory Server base DN:

```
(targetattr!="userPassword")(version 3.0; acl "Allow anonymous
read access for anyone"; allow (read,search,compare) userdn="ldap:///
anyone"
```

1. Download and unzip the Delegated Admin application package to a location on the PingDirectory Server.
2. Copy the Delegated Admin delegator folder and its contents into the PingDirectory Server's `webapps` directory.
3. In the Delegated Admin application directory, copy or rename the `example.config.js` file to `config.js`.
4. In an editor, open the `config.js` and change the values for the variables according to the local setup. The file contains comments and placeholders for necessary information. Save the changes.

   The client ID needed in this file is the one defined for the PingFederate configuration. In this file, it is the client intended for token issuance, such as `dadmin`.
5. In the PingDirectory Server root directory, run the following script to generate a `dsconfig` batch file:

   ```
   $ sh /webapps/delegator/set-up-delegator.sh
   ```

   The client ID and secret required for this script is the client configured for token verification on the PingFederate server, such as `pingdirectory`.
6. Apply the commands in the generated batch file to the PingDirectory Server. If installing in a replicated topology, add the `--applyChangeTo server-group` argument.

   ```
   $ ./bin/dsconfig \
     --bindDN "cn=Directory Manager" \
     --no-prompt \
     --batch-file webapps/delegator/delegated-admin.dsconfig -w
    directoryPassword
   ```
7. Go to `https://directoryHost:httpPort/delegator` to log in to the application.

Upgrade Delegated Admin application

If upgrading the Delegated Admin application from a previous version, unzip the new zip file and copy the delegator folder into the PingDirectory Server `webapps` folder, merging with the previous files and removing any files no longer in the new zip file. Make sure that the `config.js` file (which was customized during the first installation) is still present.

This version of the Delegated Admin application requires PingDirectory Server, version 7.0.1.

Install on PingDirectoryProxy Server

The Delegated Admin application can be installed on PingDirectoryProxy Server instances using the same PingDirectory Server installation instructions and setup script. Although it is not necessary to run the Delegated Admin application on the PingDirectory Server instances behind PingDirectoryProxy Server, all of the PingDirectory Server instances must have the following configuration elements exactly as they are configured on PingDirectoryProxy Server:

- The "Delegated Admin Privilege" virtual-attribute.
- The `delegated-admin-resource-type`, including all referenced `delegated-administrator` and `delegated-group-administrator` definitions.
- The global ACI "Authenticated access to the multi-update extended request for the Delegated Admin API."
- The global ACI "Authenticated access to the no-op request control for the Delegated Admin API."

The following configuration elements are not needed on the PingDirectory Server instances behind PingDirectoryProxy Server:

- The "Delegator" web-application-extension.
- The "PingFederateValidator" access-token-validator.

These sections must be removed from the script before the script is run on the backend PingDirectory Server instances.

Configure delegated administrators on the PingDirectory Server

To use the Delegated Admin application, an administrator must have more than valid credentials and an access token successfully validated by the PingDirectory Server. The administrator must be designated through the Directory Server configuration. To delegate users or groups as administrators, use the PingDirectory Server Administrative Console (Delegated Admin Resource Types), or the `dsconfig create-delegated-administrator` command.

The following sample commands illustrate configuration options for delegated administration, and are performed on the PingDirectory Server.

- The following command restricts an administrator to manage specified subtrees:

```
$ bin/dsconfig create-delegated-administrator \
  --type-name users \
  --administrator-name admin1 \
  --set "admin-user-dn:uid=admin1,ou=people,dc=example,dc=com" \
  --set admin-scope:manages-specific-subtrees \
  --set "managed-subtree:ou=org1,dc=example,dc=com" \
  --set enabled:true
```

- An administrator could be restricted to manage the member users of one or more specified groups. The following example assumes the existence of a static or dynamic group entry whose members include the users to be managed.

```
$ bin/dsconfig create-delegated-administrator \
  --type-name users \
  --administrator-name admin1 \
  --set "admin-user-dn:uid=admin1,ou=people,dc=example,dc=com" \
  --set admin-scope:manages-users-in-specific-groups \
  --set "managed-users-in-group:cn=User Group,dc=example,dc=com" \
  --set enabled:true
```

- Rather than delegate a single user as an administrator, it may be more convenient to delegate an entire group of users as administrators:

```
$ bin/dsconfig create-delegated-administrator \
  --type-name users \
  --administrator-name admin-group1 \
  --set "admin-group-dn:cn=Admin Group,ou=people,dc=example,dc=com" \
  --set admin-scope:manages-all-entries \
  --set enabled:true
```

  In this example, groups can be configured to use manage specific subtrees or groups with the `manages-specific-subtrees` or `manages-users-in-specific-groups` settings for the `admin-scope`. For more information about PingDirectory Server administrators and configuring dynamic and static groups, see the *PingDirectory Server Administration Guide*.

Configure attributes and attribute search on PingDirectory Server

The file used to install the Delegated Admin application specifies the object class of user entries through `structural-ldap-objectclass:inetOrgPerson`, and specifies a number of user attributes to be exposed by the application.

1. If needed, the attribute designated as the primary display attribute can be changed.

```
$ bin/dsconfig set-delegated-admin-resource-type-prop \
  --type-name users \
  --set primary-display-attribute-type:mail
```

2. Configure any additional user attributes to display in the Delegated Admin application by specifying the LDAP attribute type to be exposed and providing a display name for it.

   ```
   $ bin/dsconfig create-delegated-admin-attribute \
     --type-name users \
     --attribute-type customAttr
     --set "display-name:My custom attribute"
   ```

3. When search text is entered in the application, the attributes to be searched in the PingDirectory Server are specified by the `search-filter-pattern` property. The PingDirectory Server must have appropriate attribute indices defined to satisfy the query. Use the following command to set the search filter.

   ```
   $ bin/dsconfig set-delegated-admin-resource-type-prop \
     --type-name users \
     --set 'search-filter-pattern:(|(cn=*%%*)(mail=%%*)(uid=%%*))'
   ```

Manage groups

Group administrators can be configured to manage the membership of static groups that exist in the PingDirectory Server. The Delegated Admin application cannot create or delete these groups.

Use the following command to delegate a user as a group administrator:

```
$ bin/dsconfig create-delegated-group-administrator \
 --type-name users \
 --administrator-name group-admin1
 --set enabled:true \
 --set "admin-user-dn:uid=admin1,ou=people,dc=example,dc=com" \
 --set user-scope:users-in-specific-subtrees \
 --set "user-subtree:ou=org1,dc=example,dc=com"  \
 --set "manage-membership-of-group:cn=Employees,dc=example,dc=com" \
 --set "manage-membership-of-group:cn=Contractors,dc=example,dc=com"
```

The user scope determines which users are visible to this group administrator. In this example, all users in the subtree `ou=org1,dc=example,dc=com` are visable. An administrator can be configured as both a delegated administrator able to edit users, and also a delegated group administrator to manage group membership and view users.

The group administrator is able to view, and add or remove any of the users within the `user-scope` to the membership of groups specified by `manage-membership-of-group`. Static groups can be nested. Users belonging indirectly to a group through nesting, are visible as group members but cannot be removed. Users can only be removed from the group in which they are a member. For example, an Employees group may include a Developers group as a nested member. A user in the Developers group is a direct member of that group, and is also an indirect member of Employees. This member can only be removed when viewing the Developers group, not when viewing the Employees group.

If `manage-membership-of-group` references a dynamic group or virtual static group, rather than a static group, then that group and its members will be visible, but the group membership cannot be modified.

Set group attributes

The default settings for group attributes specify `cn` and `description` as group attributes, with `cn` used for the group title in the Delegated Admin application. The equivalent commands to create the default settings are:

```
$ bin/dsconfig create-delegated-admin-group-attribute \
  --type-name users \
  --attribute-type cn \
```

```
  --set "display-name:Name"
```

```
$ bin/dsconfig dsconfig create-delegated-admin-group-attribute \
  --type-name users \
  --attribute-type description \
  --set "display-name:Description"
```

```
$ bin/dsconfig set-delegated-admin-resource-type-prop \
  --type-name users \
  --set group-title-attribute-type:cn
```

Set group search filter

When entering search text to search for groups, the attributes to be searched in the PingDirectory Server are specified by the `group-search-filter-pattern` property. The PingDirectory Server must have appropriate attribute indices defined to satisfy the query. The default setting searches the `cn` attribute for the search text (represented by %%). Use the following command to set the group search filter:

```
$ bin/dsconfig set-delegated-admin-resource-type-prop \
  --type-name users \
  --set 'group-search-filter-pattern:(cn=*%%*)'
```

Enable log tracing

Log tracing can be enabled for OAuth token processing, HTTP request and response actions, and API debugging.

To see OAuth token processing and full HTTP request/response tracing, enable the debug trace logger as follows:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name 'Debug Trace Logger' \
  --set enabled:true
```

To enable `≈dadmin` API debug logging, use the following commands:

```
$ bin/dsconfig create-debug-target \
  --publisher-name 'File-Based Debug Logger' \
  --target-name com.unboundid.directory.server.http \
  --set debug-level:VERBOSE
```

```
$ bin/dsconfig create-debug-target \
   --publisher-name 'File-Based Debug Logger' \
   --target-name com.unboundid.directory.server.extensions.dadmin \
   --set debug-level:VERBOSE
```

```
$ bin/dsconfig create-debug-target \
  --publisher-name 'File-Based Debug Logger' \
  --target-name com.unboundid.directory.broker.api \
  --set debug-level:VERBOSE
```

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name 'File-Based Debug Logger' \
  --set enabled:true
```

Appendix A

Sample PingFederate configuration

Topics:

PingFederate offers many configuration options. This section provides a sample configuration that can be used to support the Delegated Admin application.

PingFederate sample configuration

The following is a sample PingFederate configuration. Minimal support for the Delegated Admin applications includes configuring PingFederate to use the HTML Form Adapter to authenticate users for the Delegated Admin application, and configuring PingFederate to identify an authenticated user through the user's entryUUID, which is mapped to the `subject` of the OIDC token.

Configure PingFederate as the Identity Provider

This procedure configures the PingFederate Server as the identity provider for PingDirectory Server.

Note: Before starting, download the LDAPS certificate from PingDirectory Server. All other steps are performed on the PingFederate server. See the *PingDirectory Server Administration Guide* for details.

1. Add roles for OAuth and OIDC and Identity Provider under **Server Configuration > Server Settings > Roles and Protocols**.
2. Upload the PingDirectory Server LDAPS certificate in **Server Configuration > Trusted CAs**.
3. Add an LDAP data store in **Server Configuration > Data Stores**. Specify:
   a) The PingDirectory Server hostname and LDAPS port.
   b) Select **Use LDAPS**.
   c) Under **Advanced**, clear the **Verify LDAPS hostname** option.
4. Create the HTML form **IdP Adapter and Password Credential Validator** that is used to authenticate users against PingDirectory Server:
   a) Select **Identity Provider > Adapters > Create New Instance**.
   b) Select the HTML Form type.
   c) Scroll to bottom, and click **Manage Password Credential Validators**.
   d) Select **Create New Instance**.
   e) Select the LDAP Username Password Credential Validator.
   f) To use either email address or username to log in, enter the following search filter:

   ```
   (|(uid=${username})(mail=${username}))
   ```

   g) Extend the contract with "entryUUID" and "cn". These are used later.
   h) Click **Next** and **Save** until the Create Adapter Instance screen.
   i) Add a new row to Password Credential Validators, and choose the new LDAP Password Credential Validator.
   j) Extend the contract with "entryUUID" and "cn". These are used later.
   k) Check "entryUUID" for pseudonym, and **Save**.
5. Enable session tracking in **Identity Provider > Sessions**, and select the **Track adapter session for logout** and **Enable authentication sessions for all sources** options.
6. Click **Save**.

Configure the OAuth server

1. Click **IdP Adapter Mapping**, and add the new IdP adapter for creating OAuth grants. An additional attribute source is not needed. Fulfill the contract with the USER_KEY from Adapter "entryUUID" and USER_NAME from Adapter "cn", and click **Save**.
2. Click **Access Token Management > Create New Instance**. In this step, JSON Web Tokens are configured as an example:
   a) Select **JSON Web Tokens**.
   b) Choose one-way encryption for JWT, which only requires a symmetric key (not a certificate and private key). This requires the client to validate the token by hitting the validation endpoint on the server.
   c) Add a row to symmetric keys and use 32 bytes or 64 chars of hex.
   d) Choose the JWS Algorithm HMAC using SHA-256.
   e) Choose your symmetric key for Active Symmetric Key ID, and click **Next**.

   f) Check all options.
   g) List at least one attribute to be defined in the access token. Add `sub`, and click **Save**.

3. Click **Access Token Mapping** and map the access token attributes from the persistent grant.
   a) Choose **Default Context** and the new Access Token Manager.
   b) Map the `sub` claim from the USER_KEY of the persistent grant, which is mapped from the entryUUID.
4. Click **OpenID Connect Policy Management > Add Policy**.
   a) Choose the previously created Access Token Manager.
   b) Delete all of the extended contract attributes, except `sub`. Other scopes defined, if configured.
   c) Click **Next** to reach Contract Fulfillment.
   d) Fulfill the OIDC contract `sub` with the Access Token attribute `sub`.
   e) If a default OIDC policy is not already defined, set this new policy as the default, and click **Save**.
5. Add scopes for PingDirectory Server APIs.
   a) Click **Scope Management > Exclusive Scopes**.
   b) Add a value and description for `urn:pingidentity:directory-delegated-admin`.
   c) Click **Save**.

Configure the Delegated Admin application and PingDirectory Server as OAuth clients

The following steps configure PingDirectory Server as the token validator, and the Delegated Admin application as a new client.

1. Click **Create new client**, and set the client ID and name as PingDirectory Server.
   a) Use client secret for authentication and generate a new secret. Copy the key.
   b) Check **Access Token Validation** under Allowed Grant Types, and click **Save**.

      When creating a PingFederate Access Token Validator in PingDirectory Server, use the pingdirectory client ID and secret. An Identity Mapper in will also be used in the PingDirectory Server that will take the `sub` claim and match it against the entryUUID attribute.
2. Click **Create new client**, and set the client ID and name as dadmin.
   a) Do not configure authentication.
   b) Define the redirect URI as `"https://${directoryServer:port}/delegator/*"`, with the host and port of the PingDirectory Server.
   c) Check **Bypass Authorization Approval**, and choose the Implicit Grant type.
   d) Check **Allow Exclusive Scopes** and check `urn:pingidentity:directory-delegated-admin`.
   e) Check **Implicit** grant type.
   f) Choose the OIDC policy previously created, and click **Save**.
   g) Add `"https://${directoryServer:port}"` to the CORS origins under **OAuth Server > Authorization Server Settings > Cross-Origin Settings**.

Index

PingDataSync™

Release 7.0.1

Server Administration Guide

PingDataSync Server™ Product Documentation

Trademarks

Ping Identity, the Ping Identity logo, PingFederate, PingAccess, and PingOne are registered trademarks of Ping Identity Corporation ("Ping Identity"). All other trademarks or registered trademarks are the property of their respective owners.

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Table of Contents

-

Chapter 1: Introduction

The PingDataSync Server is a high-capacity, high-reliability data synchronization and transfer pipe between source and destination topologies.

This chapter presents a general overview of the PingDataSync Server process and examples for use.

Topics include:

Overview of the PingDataSync Server

The PingDataSync Server is an efficient, Java-based server that provides high throughput, low-latency, and bidirectional real-time synchronization between two endpoint topologies consisting of PingDirectory Servers, PingDirectoryProxy Servers, PingOne, and/or Relational Database Management Systems (RDBMS) systems. The PingDataSync Server uses a dataless approach that synchronizes changes directly from the data sources in the background, so that applications can continue to update their data sources directly. The PingDataSync Server does not store any data from the endpoints themselves, thereby reducing hardware and administration costs. The server's high-availability mechanisms also makes it easy to fail over from the main PingDataSync Server to redundant instances.

Designed to run with little administrative maintenance, the PingDataSync Server includes the following features:

- High performance and availability with built-in redundancy.
- Dataless virtual architecture for a small-memory footprint and easy maintenance.
- Hassle-free setup that enables mapping attribute names, values, and DNs between endpoints. For directory server endpoints, this enables making schema and Directory Information Tree (DIT) changes without custom coding and scripting.
- Multi-vendor directory server support including the PingDirectory Server, PingDirectoryProxy Server, Nokia 8661 Directory Server, Nokia 8661 Directory Proxy Server, Oracle/Sun Directory Server Enterprise Edition, Oracle/Sun Directory Server, Oracle Unified Directory, OpenDJ, and Microsoft Active Directory, and generic LDAP directories.
- RDBMS support including Oracle Database, and Microsoft SQL Server systems.
- Proxy Server support including the PingDirectoryProxy Server and the Nokia 8661 Directory Proxy Server.
- Notification support that allows real-time change notifications to be pushed to client applications or services as they occur.

Data synchronization process

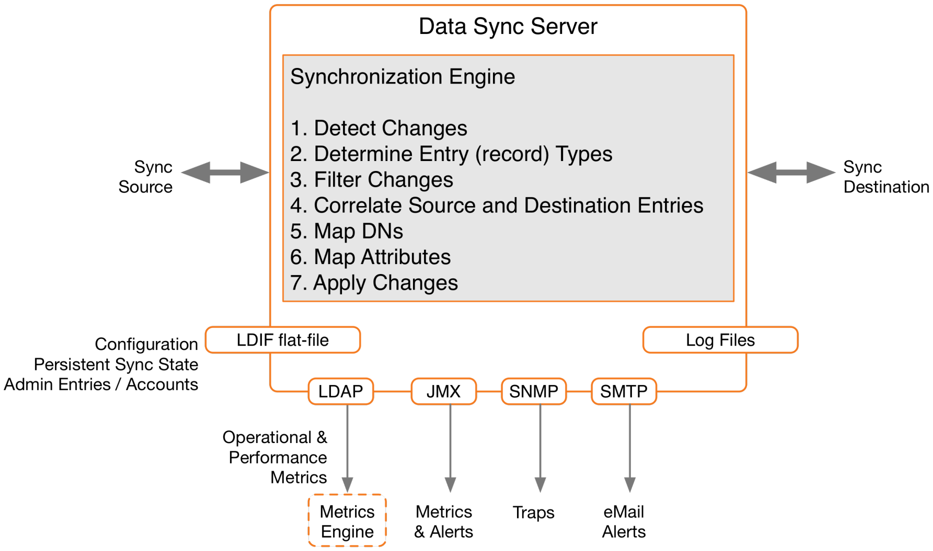
The PingDataSync Server performs point-to-point synchronization between a source endpoint and a destination endpoint. An endpoint is defined as any source or destination topology of directory or database servers.

The PingDataSync Server synchronizes data in one direction or bidirectionally between endpoints. For example, in a migration phase from Sun Directory Server to a PingData PingDirectory Server, synchronization can occur in one direction from the source server to a staging server. With one-way synchronization, the source server is the authoritative endpoint for changes in the system. Bidirectional synchronization allows for parallel active installations between the source and the destination endpoints. With bidirectional synchronization, both endpoints are authoritative for the same set of attributes or for different sets of data.

ThePingDataSync Server also contains no single point of failure, either for detecting changes or for applying changes. PingDataSync Server instances themselves are redundant. There can be multiple instances running at a time, but only the server with the highest priority is actively synchronizing changes. The stand-by servers are constantly polling the active server instance to update their persistent state. This state contains the minimum amount of information needed to begin synchronization where the primary server left off, which logically is the last processed change number for the source server. In the case of a network partition, multiple servers can synchronize simultaneously without causing problems as they each verify the full entry before making changes.

Synchronization architecture

The PingDataSync Server uses a virtualized, dataless approach that does not store directory data locally. The log files, administrator entries, configuration, sync state information are stored as flat files (LDIF format) within the system. No additional database is required.



Synchronization Architecture

Change tracking, monitoring, and logging

The PingDataSync Server tracks and manages processes and server health with the following tools:

- **Change Tracking** – Each directory instance stores a separate entry under cn=changelog for every modification made to the directory. The PingDataSync Server provides full control over the synchronization process by determining which entries are synchronized, how they are correlated to the entries at the destination endpoint, and how they are transformed into the destination schema.

- For the PingData PingDirectory Server or Nokia 8661 Directory Server topologies, the PingDataSync Server uses the servers's LDAP Change Log for modification detection.
- For Oracle/Sun Directory Server, OpenDJ, Oracle Unified Directory, and generic LDAP directory topologies, the PingDataSync Server uses the server's Retro Change Log, which provides a detailed summary of each change.
- For Active Directory, the PingDataSync Server uses the DirSync control, which polls for object attribute changes.
- For RDBMS systems, the PingDataSync Server uses an Ping Identity Server SDK plug-in to interface with a customized RDBMS change log table. Database triggers on each table record all INSERT, UPDATE, and DELETE operations to the change log table.

- **Monitoring, Alerts, and Alarms** – The PingDataSync Server supports several industry-standard, administrative protocols for monitoring, alarms, and alerts. System alarms and gauges can be configured to determine healthy performance thresholds and the server actions taken when performance values are outside the threshold. All administrative alarms are exposed over LDAP as entries under base DN `cn=alarms`. An administrative alert framework sends warnings, errors, or other server events through log messages, email, or JMX notifications. Administrative alerts are also exposed over LDAP as entries below base DN `cn=alerts`. Typical alert events are startup or shutdown, applied configuration changes, or synchronized resources unavailable.
- **Logging** – The PingDataSync Server provides standard logs (`sync`, `access`, `error`, `failed-operations`, `config-audit.log`, `debug`). The server can also be configured for multiple active sync logs. For example, each detected change, each dropped change, each applied change, or each failed change can be logged.

Synchronization Modes

The PingDataSync Server runs as a standalone Java process with two synchronization modes: standard and notification.

Standard Synchronization

In standard synchronization mode, thePingDataSync Server polls the directory server change log for create, modify, and delete operations on any entry. The server fetches the full entries from both the source and destination endpoints, and compares them to produce the minimal set of changes required to synchronize the destination with the source.

The following shows the standard synchronization change flow between two servers. The changes are processed in parallel, which increases throughput and offsets network latency.

Notification Synchronization

In notification synchronization mode, the PingDataSync Server skips the fetch and compare phases of processing and simply notifies the destination that a change has happened and provides the details of the change. Notification mode is currently available for the PingData and Alcatel-Lucent 8661 directory and proxy servers only.

PingDataSync Server Operations

The PingDataSync Server provides seamless integration between disparate systems to transform data using attribute and DN mappings. A bulk resynchronization operation can be run verify mappings and test synchronization settings.

Real-Time Synchronization

Real-time synchronization is performed with the `realtime-sync` utility. The `realtime-sync` utility polls the source server for changes and synchronizes the destination entries immediately. Once the server determines that a change should be synchronized, it fetches the full entry from the source. It then searches for the corresponding entry in the destination endpoint using correlation rules and applies the minimum set of changes to synchronize the attributes. The server fetches and compares the full entries to make sure it does not synchronize any old data from the change log.

After a synchronization topology is configured, run `resync` to synchronize the endpoints, and then run `realtime-sync` to start global synchronization.

The `realtime-sync` tool is used for the following tasks:

- Start or stop synchronization globally or for specific sync pipes only.
- Set a start point at which synchronization should begin such as the beginning or end of the change log, at a specified change number, at a specified change sequence number, or at a specified time frame in the change log.

Data Transformations

Data transformations alter the contents of synchronized entries between the source and destination directory server to handle variances in attribute names, attribute values, or DN structures. When entries are synchronized between a source and a destination server, the contents of these entries can be changed using attribute and DN mappings, so that neither server needs be aware of the transformations.

- **Attribute Mapping** – Any attribute in the entry can be renamed to fit the schema definitions from the source endpoint to the destination endpoint. This mapping makes it possible to synchronize information stored in one directory's attribute to an attribute with a different name in another directory server, or to construct an attribute using portions of the source attribute values.

- **DN Mapping** – Any DNs referenced in the entries can be transparently altered. This mapping makes it possible to synchronize data from a topology that uses one DIT structure to a system that uses a different DIT structure.

Bulk Resync

The `resync` tool performs a bulk comparison of entries on source topologies and destination topologies. The PingDataSync Server streams entries from the source, and either updates the corresponding destination entries or reports those that are different. The `resync` utility resides in the `/bin` folder (UNIX or LINUX) or `\bat` folder (Windows), and can be used for the following tasks:

- Verify that the two endpoints are synchronized after an initial configuration.
- Initially populate a newly configured target endpoint.
- Validate that the server is behaving as expected. The `resync` tool has a `--dry-run` option that validates that synchronization is operating properly, without updating any entries. This option also can be used to check attribute or DN mappings.
- Perform scheduled synchronization.
- Recover from a failover by resynchronizing entries that were modified since the last backup was taken.

The `resync` tool also enables control over what can be synchronized, such as:

- Include or exclude any source and destination attributes.
- Apply an LDAP filter to only sync entries created since that last time the tool ran.
- Synchronize only creations or only modifications.
- Change the logging verbosity.
- Set a limit on `resync` operations (such as 2000 operations per second) to reduce impact on endpoint servers.

The Sync Retry Mechanism

The PingDataSync Server is designed to quickly synchronize data and attempt a retry should an operation fail for any reason. The retry mechanism involves two possible retry levels, which are configurable on the Sync Pipe configuration using advanced Sync Pipe properties. For detailed information, see the *PingDataSync Server Reference Guide* for the Sync Pipe configuration parameters.

Retry involves two possible levels:

First Level Retry – If an operation fails to synchronize, the server will attempt a configurable number of retries. The total number of retry attempts is set in the `max-operation-attempts` property on the Sync Pipe. The property indicates how many times a worker thread should retry the operation before putting the operation into the second level of retry, or failing the operation altogether.

Second Level Retry – Once the `max-operation-attempts` property has been exceeded, the retry is sent to the second level, called the delayed-retry queue. The delayed-retry queue uses two advanced Sync Pipe properties to determine the number of times an operation should be retried in the background after a specified delay.

Operations that make it to this level will be retried after the `failed-op-background-retry-delay` property (default: 1 minute). Next, the PingDataSync Server checks the `max-failed-op-background-retries` property to determine the number of times a failed operation should be retried in the background. By default, this property is set to `0`, which indicates that no background retry should be attempted, and that the operation should be logged as failed.

> **Note**
> Background operations can hold up processing other changes, since the PingDataSync Server will only process up to the next 5000 changes while waiting for a retried operation to complete.

Retry can be controlled by the custom endpoint based on the type of error exception. When throwing an exception, the endpoint code can signal that a change should be aborted, retried a limited number of times, or retried an unlimited number of times. Some errors, such as endpoint server down, should be retried indefinitely.

If the `max-failed-op-background-retries` property has been exceeded, the retry is logged as a failure and appears in the `sync` and the `sync-failed-ops` logs.

Configuration Components

The PingDataSync Server supports the following configuration parameters that determine how synchronization takes place between directories or databases:

Sync Pipe – Defines a single synchronization path between the source and destination topologies. Every Sync Pipe has one or more Sync Classes that control how and what is synchronized. Multiple Sync Pipes can run in a single server instance.

Sync Source – Defines the directory topology that is the source of the data to be synchronized. A Sync Source can reference one or more supported external servers.

Sync Destination – Defines the topology of directory servers where changes detected at the Sync Source are applied. A Sync Destination can reference one or more external servers.

External Server – Defines a single server in a topology of identical, replicated servers to be synchronized. A single external server configuration object can be referenced by multiple Sync Sources and Sync Destinations.

Sync Class – Defines the operation types and attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated. A source entry is in one Sync Class and is determined by a base DN and LDAP filters. A Sync Class can reference zero or more Attribute Maps and DN Maps, respectively. Within a Sync Pipe, a Sync Class is defined for each type of entry that needs to be treated differently. For example, entries that define attribute mappings, or entries that should not be synchronized at all. A Sync Pipe must have at least one Sync Class but can refer to multiple Sync Class objects.

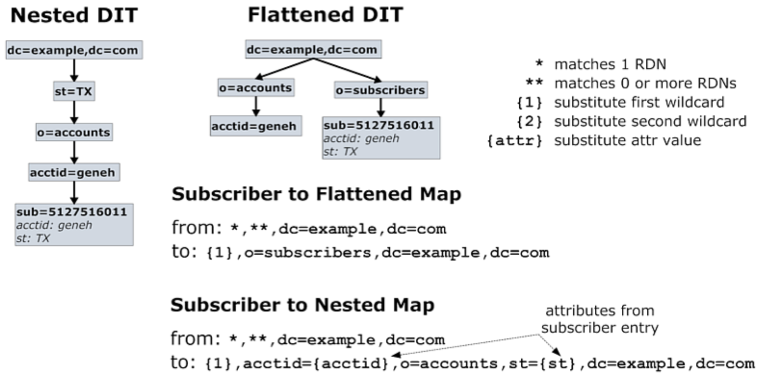
DN Map – Defines mappings for use when destination DNs differ from source DNs. These mappings allow the use of wild cards for DN transformations. A single wild card ("*") matches a single RDN component and can be used any number of times. The double wild card ("**") matches zero or more RDN components and can be used only once. The wild card values can be used in the `to-dn-pattern` attribute using `{1}` and their original index position in the pattern, or `{attr}` to match an attribute value. For example:

```
**,dc=myexample,dc=com->{1},o=example
```

Regular expressions and attributes from the user entry can also be used. For example, the following mapping constructs a value for the `uid` attribute, which is the RDN, out of the initials (first letter of `givenname` and `sn`) and the employee ID (`eid` attribute).

```
uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid},{2},o=example
```

The following illustrates a how a nested DIT can be mapped to a flattened structure.


Nested DIT
dc=example,dc=com
st=TX
o=accounts
acctid=geneh
sub=5127516011
acctid: geneh
st: TX
Flattened DIT
dc=example,dc=com
o=accounts
o=subscribers
acctid=geneh
sub=5127516011
acctid: geneh
st: TX
* matches 1 RDN
** matches 0 or more RDNs
{1} substitute first wildcard
{2} substitute second wildcard
{attr} substitute attr value
Subscriber to Flattened Map
from: *,**,dc=example,dc=com
to: {1},o=subscribers,dc=example,dc=com
Subscriber to Nested Map
attributes from subscriber entry
from: *,**,dc=example,dc=com
to: {1},acctid={acctid},o=accounts,st={st},dc=example,dc=com

Nested DIT Mapping

Attribute Map and Attribute Mappings – Defines a mapping for use when destination attributes differ from source attributes. A Sync Class can reference multiple attribute maps. Multiple Sync Classes can share the same attribute map. There are three types of attribute mappings:

- Direct Mapping – Attributes are directly mapped to another attribute: For example:

  ```
  employeenumber->employeeid
  ```

- Constructed Mapping – Destination attribute values are derived from source attribute values and static text. For example:

  ```
  {givenname}.{sn}@example.com->mail
  ```

- DN Mapping – Attributes are mapped for DN attributes. The same DN mappings that map entry DNs can be referenced. For example:

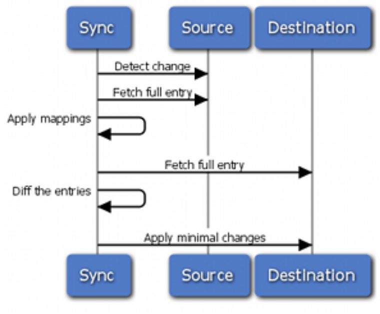
  ```
  uid=jdoe,ou=People,dc=example,dc=com.
  ```

Sync Flow Examples

The PingDataSync Server processes changes by fetching the most up-to-date, full entries from both sides and then compares them. This process flow is called standard synchronization mode. The processing flow differs depending on the type of PingDataSync Serverchange (`ADD`, `MODIFY`, `DELETE`, `MODDN`) that is requested. The following examples show the control flow diagrams for the sync operations, especially for those cases when a `MODIFY` or a `DELETE` operation is dropped. The sync log records all completed and failed operations.

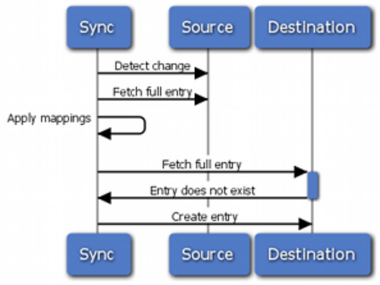
Modify Operation Example

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. Diff the computed destination entry and actual destination entry.
6. Apply the changes to the destination.



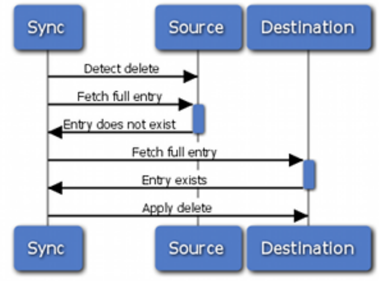
Add Operation Example

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. The entry or table row does not exist on the destination.
6. Create the entry or table row.



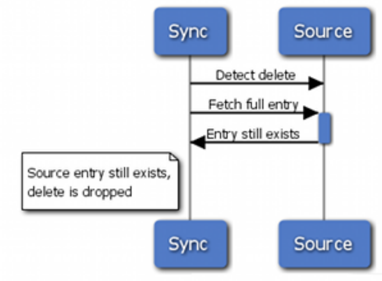
Delete Operation Example

1. Detect delete from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. The entry or table row exists on the destination.
6. Apply the delete on the destination.



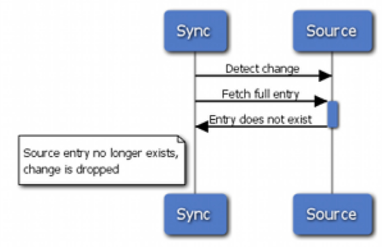
Delete After Source Entry is Re-Added

1. Detect delete from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. The entry or table row exists on the source.
4. Delete request is dropped.



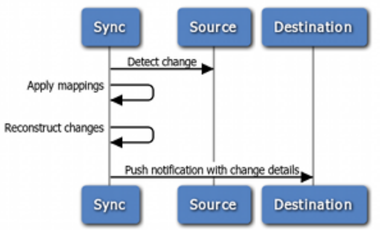
Standard Modify After Source Entry is Deleted

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. The entry does not exist.
4. Change request is dropped because the source entry no longer exists.



Notification Add, Modify, ModifyDN, and Delete

1. Detect change from the change log table on the source.
2. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
3. Reconstruct changed entries.
4. Push notification with change details to the destination.



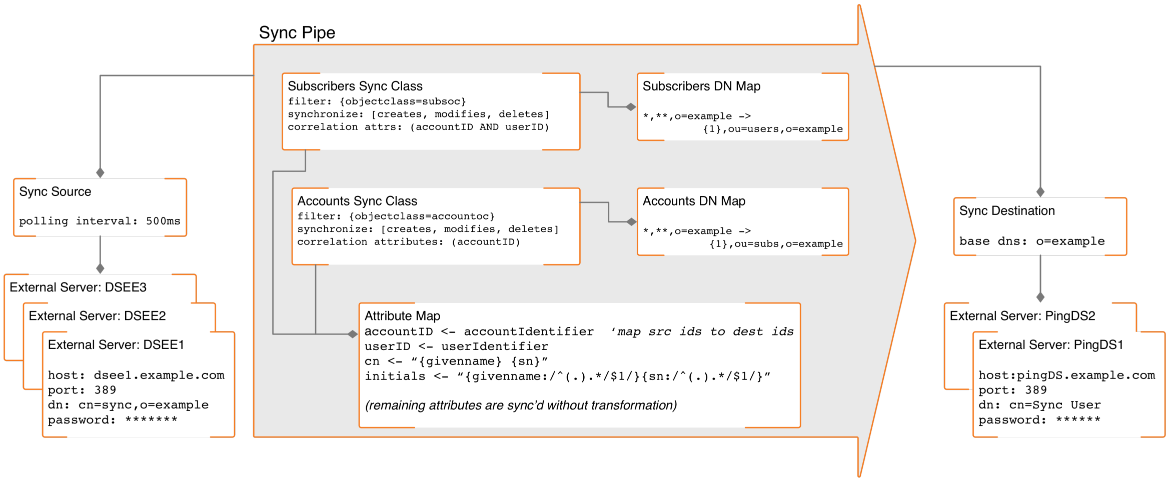
Sample Synchronization

The following is a synchronization migration example from a Sun Directory Server Enterprise Edition (DSEE) topology to the PingData PingDirectory Server topology, including a change in the DIT structure to a flattened directory structure. The Sync Pipe connects the Sun Directory Server topology as the Sync Source and the PingDirectory Server topology as the Sync Destination. Each endpoint is defined with three external servers in their respective topology. The Sync Pipe destination has its base DN set to `o=example`, which is used when performing LDAP searches for entries.

Two Sync Classes are defined: one for Subscribers, and one for Accounts. Each Sync Class uses a single "Sun DS to PingData Attribute Map" that has four attribute mappings defined. Each Sync Class also defines its own DN maps. For example, the Accounts Sync Class uses a DN map, called PingData Account Map, that is used to flatten a hierarchical DIT, so that the Account entries appear directly under `ou=accounts` as follows:

```
*,**,o=example -> {1},ou=accounts,o=example
```

With this mapping, if an entry DN has `uid=jsmith,ou=people,o=example`, then "`*`" matches `uid=jsmith`, "`**`" matches `ou=people`, and `{1}` matches `uid=jsmith`. Therefore, `uid=jsmith,ou=people,o=example` gets mapped to `uid=jsmith,ou=accounts,o=example`. A similar map is configured for the Subscribers Sync Class.



```
filter: {objectclass=subsoc}
synchronize: [creates, modifies, deletes]
correlation attrs: (accountID AND userID)

*,**,o=example ->
        {1},ou=users,o=example

filter: {objectclass=accountoc}
synchronize: [creates, modifies, deletes]
correlation attributes: (accountID)

*,**,o=example ->
         {1},ou=subs,o=example

accountID <- accountIdentifier  'map src ids to dest ids
userID <- userIdentifier
cn <- "{givenname} {sn}"
initials <- "{givenname:/^(.).*/$1/}{sn:/^(.).*/$1/}"

(remaining attributes are sync'd without transformation)

polling interval: 500ms

host: dsee1.example.com
port: 389
dn: cn=sync,o=example
password: *******

base dns: o=example

host:pingDS.example.com
port: 389
dn: cn=Sync User
password: ******
```

A Sample Synchronization Topology Configuration

Chapter 2: Installing the PingDataSync Server

This section describes how to install and run the PingDataSync Server. It includes pre-installation requirements and considerations.

Topics include:

Supported Platforms

The PingDataSync Server is a pure Java application. It is intended to run within the Java Virtual Machine on any Java Standard Edition (SE) or Enterprise Edition (EE) certified platform. For the list of supported platforms and Java versions, access the Ping Identity Customer Support Center portal or contact a PingData authorized support provider.

> **Note**
> It is highly recommended that a Network Time Protocol (NTP) system be in place so that multi-server environments are synchronized and timestamps are accurate.

Install the JDK

The Java 64-bit JDK is required on the server. Even if Java is already installed, create a separate Java installation for use by the server to ensure that updates to the system- wide Java installation do not inadvertently impact the installation.

Optimize the Linux Operating System

Configure the Linux filesystem by making the following changes.

> **Note**
> The server explicitly overrides environment variables like `PATH`, `LD_LIBRARY_PATH`, and `LD_PRELOAD` to ensure that settings used to start the server do not inadvertently impact its behavior. If these variables must be edited, set values by editing the `set_environment_vars` function of the `lib/_script-util.sh` script. Stop and restart the server for the change to take effect.

Set the file descriptor limit

The server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. If needed, increase the file descriptor limit on the operating system with the following procedure.

> **Note**
> If the operating system relies on `systemd`, refer to the Linux operating system documentation for instructions on setting the file descriptor limit.

1. Display the current hard limit of the system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

   ```
   ulimit -aH
   ```

2. Edit the `/etc/sysctl.conf` file. If the `fs.file-max` property is defined in the file, make sure its value is set to at least 65535. If the line does not exist, add the following to the end of the file:

   ```
   fs.file-max = 65535
   ```

3. Edit the /etc/security/limits.conf file. If the file has lines that set the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before #End of file). Insert a tab between the columns.

   ```
   * soft nofile 65535
   * hard nofile 65535
   ```

4. Reboot the system, and then use the ulimit command to verify that the file descriptor limit is set to 65535 with the following command:

   ```
   ulimit -n
   ```

Once the operating system limit is set, the number of file descriptors that the server will use can be configured by either using a NUM_FILE_DESCRIPTORS environment variable, or by creating a config/num-file-descriptors file with a single line such as, NUM_FILE_DESCRIPTORS=12345. If these are not set, the default of 65535 is used. This is strictly optional if wanting to ensure that the server shuts down safely prior to reaching the file descriptor limit.

Note
For RedHat 7 or later, modify the 20-nproc.conf file to set both the open files and max user processes limits:

```
/etc/security/limits.d/20-nproc.conf

Add or edit the following lines if they do not already exist:
*     soft    nproc     65536
*     soft    nofile    65536
*     hard    nproc     65536
*     hard    nofile    65536
root  soft    nproc     unlimited
```

Set the filesystem flushes

Linux systems running the ext3 filesystem only flush data to disk every five seconds. If the server is on a Linux system, edit the mount options to include the following:

```
commit=1
```

This variable changes the flush frequency from five seconds to one. Also, set the flush frequency in the /etc/fstab file to make sure the configuration remains after reboot.

Install sysstat and pstack on Red Hat

The server troubleshooting tool collect-support-data relies on the iostat, mpstat, and pstack utilities to collect monitoring, performance statistics, and stack trace information on the server’s processes. For Red Hat systems, make sure that these packages are installed, for example:

```
$ sudo yum install sysstat gdb dstat -y
```

Install the dstat utility

The dstat utility is used by the collect-support-data tool.

Disable filesystem swapping

Any performance tuning services, like tuned, should be be disabled. If performance tuning is required, vm.swappiness can be set by cloning the existing performance profile then adding vm.swappiness = 0 to the new profile's tuned.conf file in /usr/lib/tuned/profile-name/tuned.conf. The updated profile is then selected by running tuned-adm profile customized_profile.

Manage system entropy

Entropy is used to calculate random data that is used by the system in cryptographic operations. Some environments with low entropy may have intermittent performance issues with SSL-based communication. This is more typical on virtual machines, but can occur in physical instances as well. Monitor the kernel.random.entropy_avail in sysctl value for best results.

If necessary, update $JAVA_HOME/jre/lib/security/java.security to use file:/dev/./urandom for the securerandom.source property.

Enable the server to listen on privileged ports

Linux provides 'capabilities' used to grant specific commands the ability to do things that are normally only allowed for a root account. Instead of granting the ability to a specific user, capabilities are granted to a specific command. It may be convenient to enable the server to listen on privileged ports while running as a non-root user.

The setcap command is used to assign capabilities to an application. The cap_net_bind_service capability enables a service to bind a socket to privileged ports (port numbers less than 1024). If Java is installed in /ds/java (and the Java command to run the server is /ds/java/bin/java), the Java binary can be granted the cap_net_bind_service capability with the following command:

```
$ sudo setcap cap_net_bind_service=+eip /ds/java/bin/java
```

The java binary needs an additional shared library (libjli.so) as part of the Java installation. More strict limitations are imposed on where the operating system will look for shared libraries to load for commands that have capabilities assigned. So it is also necessary to tell the operating system where to look for this library. This can be done by creating the file /etc/ld.so.conf.d/libjli.conf with the path to the directory that contains the libjli.so file. For example, if the Java installation is in /ds/java, the contents of that file should be:

```
/ds/java/lib/amd64/jli
```

Run the following command for the change to take effect:

```
$ sudo ldconfig -v
```

Ping license keys

License keys are required to install all Ping products. Obtain licenses through Salesforce or from https://www.pingidentity.com/en/account/request-license-key.html.

- A license is always required for setting up a new single server instance and can be used site-wide for all servers in an environment. When cloning a server instance with a valid license, no new license is needed.
- A new license must be obtained when updating a server to a new major version, for example from 6.2 to 7.0. Licenses with no expiration date are valid until the server is upgraded to the next major version. A prompt for a new license is displayed during the update process.
- A license may expire on particular date. If a license does expire, obtain a new license and install it using `dsconfig` or the Administrative Console. The server will provide a notification as the expiration date approaches. License details are available using the server's `status` tool.

When installing the server, specify the license key file in one of the following ways:

- Copy the license key file to the server root directory before running setup. The interactive `setup` tool will discover the file and not require input. If the file is not in the server root, the setup tool will prompt for its location.
- If the license key is not in the server root directory, specify the `--licenseKeyFile` option for non-interactive setup, and the path to the file.

Install the PingDataSync Server

Use the `setup` tool to install the server. The server needs to be started and stopped by the user who installed it.

> **Note**
> A Windows installation requires that the Visual Studio 2010 runtime patch be installed prior to running the `setup` command.

1. Log in as a user, other than root.
2. Obtain the latest zip release bundle from Ping Identity and unpack it in a directory owned by this user.

   ```
   $ unzip PingDataSync-<version>.zip
   ```

3. Change to the server root directory.

   ```
   $ cd PingDataSync
   ```

4. Run the `setup` command.

   ```
   $ ./setup
   ```

5. Type **yes** to accept the End-User License Agreement and press **Enter** to continue.
6. If adding this server to an existing PingDataSync Server topology, type **yes**, or press **Enter** to accept the default (no).
7. Enter the fully qualified host name or IP address of the local host.
8. Create the initial root user DN for the PingDataSync Server, or press **Enter** to accept the default (cn=Directory Manager).
9. Enter and confirm a password for this account.
10. Press **Enter** to enable server services and the Administrative Console.
11. Enter the port on which the PingDataSync Server will accept connections from HTTPS clients, or press **Enter** to accept the default.
12. Enter the port on which the PingDataSync Server will accept connections from LDAP clients, or press **Enter** to accept the default.
13. Press **Enter** to enable LDAPS, or enter **no**.
14. Press **Enter** to enable StartTLS, or enter **no**.
15. Select the certificate option for this server.
16. Choose the desired encryption for the directory data, backups, and log files from the choices provided:
    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.
    - Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
    - Do not encrypt server data.
17. Choose the option for the amount of memory that should be allocated to the server.
18. To start the server when the configuration is complete, press **Enter** for (yes).
19. A Setup Summary is displayed. choose the option to setup the server with the listed parameters, change the parameters, or cancel the setup.

After the server configuration is complete, the `create-sync-pipe-config` tool can be run to configure the synchronization environment.

The PingDataSync Server Administrative Console enables browser-based server management, the `dsconfig` tool enables command line management, and the Configuration API enables management by third-party interfaces.

Log into the Administrative Console

After the server is installed, access the Administrative Console, `https://<host>/console/login`, to verify the configuration and manage the server. The root user DN or the common name of a root user DN is required to log into the Administrative Console. For example, if the DN created when the server was installed is `cn=Directory Manager`, `directory manager` can be used to log into the Administrative Console.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package can be installed according to that container's documentation. Contact Ping Identity Customer Support for the package location and instructions.

Server folders and files

After the distribution file is unzipped, the following folders and command-line utilities are available:

Directories/Files/Tools	Description
ldif	Stores any LDIF files that you may have created or imported.
import-tmp	Stores temporary imported items.
classes	Stores any external classes for server extensions.
bak	Stores the physical backup files used with the backup command-line tool.
velocity	Stores Velocity templates that define the server's application pages.
update.bat, and update	The update tool for UNIX/Linux systems and Windows systems.
uninstall.bat, and uninstall	The uninstall tool for UNIX/Linux systems and Windows systems.
ping_logo.png	The image file for the Ping Identity logo.
setup.bat, and setup	The setup tool for UNIX/Linux systems and Windows systems.
revert-update.bat, and revert-update	The revert-update tool for UNIX/Linux systems and Windows systems.
README	README file that describes the steps to set up and start the server.
License.txt	Licensing agreement for the product.
legal-notices	Stores any legal notices for dependent software used with the product.
docs	Provides the release notes, Configuration Reference (HTML), API Reference, and all other product documentation.
metrics	Stores the metrics that can be gathered for this server and surfaced in the PingDataMetrics Server.

Directories/Files/Tools	Description
bin	Stores UNIX/Linux-based command-line tools.
bat	Stores Windows-based command-line tools.
lib	Stores any scripts, jar files, and library files needed for the server and its extensions.
collector	Used by the server to make monitored statistics available to the PingDataMetrics Server.
locks	Stores any lock files in the backends.
tmp	Stores temporary files.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
logs	Stores log files.
AD-Password-Sync-Agent.zip	The Active Directory Sync Agent package.

Start and stop the server

To start the PingDataSync Server, run the `bin/start-sync-server` command on UNIX or Linux systems (the `bat` folder on Microsoft Windows systems).

Start the Server as a Background Process

Navigate to the server root directory, and run the following command:

```
$ bin/start-server
```

For Windows systems:

```
$ bat/start-server
```

Start the server at boot time

By default, the server does not start automatically when the system is booted. To configure the server to start automatically, use the `create-rc-script` tool to create a run control script as follows:

1. Create the startup script. In this example `ds` is the user.

   ```
   $ bin/create-rc-script \
     --outputFile Ping-Identity-Sync.sh \
     --userName ds
   ```

2. Log in as root, move the generated `Ping-Identity-Sync.sh` script into the `/etc/init.d` directory, and create symlinks to it from the `/etc/rc3.d` (starting with an "S" to start

the server) and `/etc/rc0.d` directory (starting with a "K" to stop the server).

```
# mv Ping-Identity-Sync.sh /etc/init.d/
# ln -s /etc/init.d/Ping-Identity-Sync.sh /etc/rc3.d/S50-Ping-Identity-
Sync.sh
# ln -s /etc/init.d/Ping-Identity-Sync.sh /etc/rc0.d/K50-Ping-Identity-
Sync.sh
```

Stop the Server

If the PingDataSync Server has been configured to use a large amount of memory, it can take several seconds for the operating system to fully release the memory. Trying to start the server too quickly after shut down can fail because the system does not yet have enough free memory. On UNIX systems, run the `vmstat` command and watch the values in the "free" column increase until all memory held by the PingDataSync Server is released back to the system.

A configuration option can also be set that specifies the maximum shutdown time a process can take.

To stop the server, navigate to the server root directory and run the following command:

```
$ bin/stop-server
```

Restart the server

Restart the server using the `bin/stop-server` command with the `--restart` or `-R` option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again.

```
$ bin/stop-server --restart
```

Run the server as a Microsoft Windows service

The server can run as a Windows service on Windows Server 2012 R2 and Windows Server 2016. This enables log out of a machine without the server being stopped.

Register the service

Perform the following steps to register the server as a service:

1. Stop the server with `bin/stop-server`. A server cannot be registered while it is running.
2. Register the server as a service. From a Windows command prompt, run `bat/register-windows-service.bat`.
3. After a server is registered, start the server from the Windows Services Control Panel or with the `bat/start-server.bat` command.

Note
Command-line arguments for the `start-server.bat` and `stop-server.bat` scripts are not supported while the server is registered to run as a Windows service. Using a task to stop the server is also not supported.

Run multiple service instances

Only one instance of a particular service can run at one time. Services are distinguished by the `wrapper.name` property in the `<server-root>/config/wrapper-product.conf` file. To run additional service instances, change the `wrapper.name` property on each additional instance. Descriptions of the services can also be added or changed in the `wrapper-product.conf` file.

Deregister and uninstall

While a server is registered as a service, it cannot run as a non-service process or be uninstalled. Use the `bat/deregister-windows-service.bat` file to remove the service from the Windows registry. The server can then be uninstalled with the `uninstall.bat` script.

Log files

The log files are stored in `<server-root>/logs`, and filenames start with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or due to file size. Only the last three log files are retained. These configurations can be changed in the `<server-root>/config/wrapper.conf` file.

Uninstall the server

Use the `uninstall` command-line utility to uninstall the server using either interactive or non-interactive modes. Interactive mode provides options, progress, and a list of the files and directories that must be manually deleted if necessary.

Non-interactive mode, invoked with the `--no-prompt` option, suppresses progress information, except for fatal errors. All options for the `uninstall` command are listed with the `--help` option.

The `uninstall` command must be run as either the root user or the user account that installed the server.

Perform the following steps to uninstall in interactive mode:

1. Navigate to the server root directory.

   ```
   $ cd PingData<server>
   ```

2. Start the uninstall command:

   ```
   $ ./uninstall
   ```

3. Select the components to be removed, or press **Enter** to remove all components.
4. If the server is running, press **Enter** to shutdown the server before continuing.

5. Manually remove any remaining files or directories, if required.

Update servers in a topology

An update to the current release includes significant changes, and the introduction of a topology registry, which will store information previously stored in the admin backend (server instances, instance and secret keys, server groups, and administrator user accounts). For the admin backend to be migrated, the `update` tool must be provided LDAP authentication options to the peer servers of the server being updated.

The LDAP connection security options requested (either plain, TLS, StartTLS, or SASL) must be configured on every server in the topology. The LDAP credentials must be present on every server in the topology, and must have permissions to read from the admin backend and the config backend of every server in thetopology. For example, a root DN user with the `inherit-default-privileges` set to true (such as the `cn=Directory Manager` user) that exists on every server can be used.

After enabling or fixing the configuration of the LDAP connection handler(s) to support the desired connection security mechanism on each server, run the following `dsframework` command on the server being updated so that its admin backend has the most up-to-date information:

```
$ bin/dsframework set-server-properties \
  --serverID serverID \
  --set ldapport:port \
  --set ldapsport:port \
  --set startTLSEnabled:true
```

The `update` tool will verify that the following conditions are satisfied on every server in the topology before allowing the update:

- When the first server is being updated, all other servers in the topology must be online. When updating additional servers, all topology information will be obtained from one of the servers that has already been updated. The `update` tool will connect to the peer servers of the server being updated to obtain the necessary information to populate the topology registry. The provided LDAP credentials must have read permissions to the config and admin backends of the peer servers.
- The instance name is set on every server, and is unique across all servers in the topology. The instance name is a server’s identifier in the topology. After all servers in the topology have been updated, each server will be uniquely identified by its instance name. Once set, the name cannot be changed. If needed, the following command can be used to set the instance name of a server prior to the update:

  ```
  $ bin/dsconfig set-global-configuration-prop \
    --set instance-name:uniqueName
  ```

- The cluster-wide configuration is synchronized on all servers in the topology. Older versions have some topology configuration under `cn=cluster,cn=config` (JSON attribute and field constraints). These items did not support mirrored cluster-wide configuration data. An update should avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. To synchronize the cluster-wide configuration data across all servers in the topology, run the config-diff tool on each pair of servers to determine the differences, and use dsconfig to update each instance using the config-diff output. For example:

```
$ bin/config-diff --sourceHost hostName \
  --sourcePort port \
  --sourceBindDN bindDN \
  --sourceBindPassword password \
  --targetHost hostName \
  --targetPort port \
  --targetBindDN bindDN \
  --targetBindPassword password
```

If any of these conditions are not satisfied, the update tool will list all of the errors encountered for each server, and provide instructions on how to fix them.

Update the server

This procedure assumes that an existing version of the server is stored at PingData-server-old. Make sure a complete, readable backup of the existing system is available before upgrading the server. Also, make sure there is a clear backout plan and schedule.

1. Download the latest version of the server software and unzip the file. For this example, the new server is located in the PingData-server-new directory.
2. Use the update tool of the newly unzipped build to update the server. Make sure to specify the server instance that is being upgrading with the --serverRoot option. The server must be stopped for the update to be applied.

Reverting an Update

If necessary, a server can be reverted to the previous version using the revert-update tool. The tool accesses a log of file actions taken by the update tool to put the filesystem back to its prior state. If multiple updates have been run, the revert-update tool can be used multiple times to revert to each prior update sequentially. For example, the revert-update command can be run to revert to the server's previous state, then run again to return to its original state. The server is stopped during the revert-update process.

Note
Reverting an update is not supported for upgrades to version 7.0, due to the topology backend changes.

Use the revert-update tool in the server root directory revert back to the most recent version of the server:

```
$ PingData-server-old/revert-update
```

Install a failover server

Ping Identity supports redundant failover servers that automatically become active when the primary server is not available. Multiple servers can be present in the topology in a configurable prioritized order.

Before installing a failover server, have a primary server already installed and configured. When installing the redundant server, the installer will copy the first server's configuration.

The primary and secondary server configuration remain identical. Both servers should be registered to the `all servers` group and all `dsconfig` changes need to be applied to the server group `all servers`.

> **Note**
> If the primary server has extensions defined, they should also be installed on any cloned or redundant servers. If extensions are missing from a secondary server, the following message is displayed during the installation:
>
> ```
> Extension class <com.server.directory.sync.MissingSyncExtension> was not
> found. Run manage-extension --install to install your extensions. Re-run
> setup to continue.
> ```

To remove a failover server, use the `uninstall` command.

1. Unpack the Ping Identity server zip build. Name the unpacked directory something other than the first server instance directory.

   ```
   $ unzip PingData<server>-<version>.zip -d <server2>
   ```

2. Navigate to the server root directory.

3. Use the `setup` tool in interactive mode in Install the Server, or in non-interactive mode as follows:

   ```
   $ ./setup --localHostName <server2>.example.com --ldapPort 7389 \
     --masterHostName <server1>.example.com --masterPort 8389 \
     --masterUseNoSecurity \
     --acceptLicense \
     --rootUserPassword password \
     --no-prompt
   ```

   The secondary server is now ready to take over as a primary server in the event of a failover. No `realtime-sync` invocations are needed for this server.

4. Verify the configuration by using the `bin/status` tool. Each server instance is given a priority index. The server with the lowest priority index number has the highest priority.

   ```
   $ bin/status --bindPassword secret

   ...(status output)...
                                 --- Sync Topology ---
   Host:Port                             :Status       :Priority
   --------------------------------------:-------------:---------------
   <server>.example.com:389 (this server) : Active      : 1
   <server>.example.com:389               : Unavailable : 2
   ```

5. Obtain the name of a particular server, run the `dsconfig` tool with the `list-external-servers` option.

   ```
   $ bin/dsconfig list-external-servers
   ```

6. To change the priority index of the server, use the `bin/dsconfig` tool:

   ```
   $ bin/dsconfig set-external-server-prop \
     --server-name <server2>.example.com:389 \
     --set <server>-priority-index:1
   ```

Administrative accounts

Users that authenticate to the Configuration API or the Administrative Console are stored in `cn=Root DNs,cn=config`. The `setup` tool automatically creates one administrative account when performing an installation. Accounts can be added or changed with the `dsconfig` tool.

Change the administrative password

Root users are governed by the Root Password Policy and by default, their passwords never expire. However, if a root user's password must be changed, use the `ldappasswordmodify` tool.

1. Open a text editor and create a text file containing the new password. In this example, name the file rootuser.txt.

   ```
   $ echo password > rootuser.txt
   ```

2. Use ldappasswordmodify to change the root user’s password.

   ```
   $ bin/ldappasswordmodify --port 1389 --bindDN "cn=Directory Manager" \
   --bindPassword secret --newPasswordFile rootuser.txt
   ```

3. Remove the text file.

   ```
   $ rm rootuser.txt
   ```

Chapter 3: Configuring the PingDataSync Server

The PingDataSync Server provides a suite of tools to configure a single server instance or server groups. All configuration changes to the server are recorded in the `config-audit.log`, and configuration is stored as a flat file (LDIF format) in the `cn=config` branch. Before configuring the PingDataSync Server, review Sync Configuration Components.

Topics include:

Configuration checklist

Prior to any deployment, determine the configuration parameters necessary for the Synchronization topology. Gather the following:

External servers

External Server Type – Determine the type of external servers included in the synchronization topology. See Overview of the PingDataSync Server for a list of supported servers.

LDAP Connection Settings – Determine the host, port, bind DN, and bind password for each external server instance(s) included in the synchronization topology.

Security and Authentication Settings – Determine the secure connection types for each external server (SSL or StartTLS). Determine authentication methods for external servers such as simple, or external (SASL mechanisms). If synchronizing passwords, encoded or especially for clear-text, the connection should be secure. If synchronizing to or from a Microsoft Active Directory system, establish an SSL or StartTLS connection to the PingDataSync Server. Password encryption should also be enabled for synchronization from Active Directory, or when synchronizing clear-text passwords.

Sync Pipes

A Sync Pipe defines a single synchronization path between the source and destination targets. One Sync Pipe is needed for each point-to-point synchronization path defined for a topology.

Sync Source – Determine which external server is the Sync Source for the synchronization topology. A prioritized list of external servers can be defined for failover purposes.

Sync Destination – Determine which external server is the Sync Destination for the synchronization topology. A prioritized list of external servers can be defined for failover purposes.

Sync Classes

A Sync Class defines how attributes and DNs are mapped and how Source and Destination entries are correlated. For each Sync Pipe defined, define one or more Sync Classes with the following information:

Evaluation Order – If defining more than one Sync Class, the evaluation order of each Sync Class must be determined with the `evaluation-order-index` property. If there is an overlap between criteria used to identify a Sync Class, the Sync Class with the most specific criteria is used first.

Base DNs – Determine which base DNs contain entries needed in the Sync Class.

Include Filters – Determine the filters to be used to search for entries in the Sync Source.

Synchronized Entry Operations – Determine the types of operations that should be synchronized: creates, modifications, and/or deletes.

DNs – Determine the differences between the DNs from the Sync Source topology to the Sync Destination topology. Are there structural differences in each Directory Information Tree (DIT)? For example, does the Sync Source use a nested DIT and the Sync Destination use a flattened DIT?

Destination Correlation Attributes – Determine the correlation attributes that are used to associate a source entry to a destination entry during the synchronization process. During the configuration setup, one or more comma-separated lists of destination correlation attributes are defined and used to search for corresponding source entries. The PingDataSync Server maps all attributes in a detected change from source to destination attributes using the attribute maps defined in the Sync Class.

The correlation attributes are flexible enough so that several destination searches with different combinations of attributes can be performed until an entry matches. For LDAP server endpoints, use the distinguished name (DN) to correlate entries. For example, specify the attribute lists `dn,uid`, `uid,employeeNumber` and `cn,employeeNumber` to correlate entries in LDAP deployments. The PingDataSync Server will search for a corresponding entry that has the same `dn` and `uid` values. If the search fails, it then searches for `uid` and `employeeNumber`. Again if the search fails, it searches for `cn` and `employeeNumber`. If none of these searches are successful, the synchronization change would be aborted and a message logged.

To prevent incorrect matches, the most restrictive attribute lists (those that will never match the wrong entry) should be first in the list, followed by less restrictive attribute lists, which will be used when the earlier lists fail. For LDAP-to-LDAP deployments, use the DN with a combination of other unique identifiers in the entry to guarantee correlation. For other non-LDAP deployments, determine the attributes that can be synchronized across the network.

Attributes – Determine the differences between the attributes from the Sync Source to the Sync Destination, including the following:

- **Attribute Mappings** – How are attributes mapped from the Sync Source to the Sync Destination? Are they mapped directly, mapped based on attribute values, or mapped based on attributes that store DN values?
- **Automatically Mapped Source Attributes** – Are there attributes that can be automatically synchronized with the same name at the Sync Source to Sync Destination? For example, can direct mappings be set for `cn`, `uid`, `telephoneNumber`, or for all attributes?
- **Non-Auto Mapped Source Attributes** – Are there attributes that should not be automatically mapped? For example, the Sync Source may have an attribute, `employee`, while the Sync Destination may have a corresponding attribute, `employeeNumber`. If an attribute is not automatically mapped, a map must be provided if it is to be synchronized.
- **Conditional Value Mapping** – Should some mappings only be applied some of the time as a function of the source attributes? Conditional value mappings can be used with the

conditional-value-pattern property, which conditionalizes the attribute mapping based on the subtype of the entry, or on the value of the attribute. For example, this might apply if the adminName attribute on the destination should be a copy of the name attribute on the source, but only if the isAdmin attribute on the source is set to true. Conditional mappings are multi-valued. Each value is evaluated until one is matched (the condition is true). If none of the conditional mappings are matched, the ordinary mappings is used. If there is not an ordinary mapping, the attribute will not be created on the destination.

The Sync User account

The PingDataSync Server creates a Sync User account DN on each external server. The account (by default, cn=Sync User) is used exclusively by the PingDataSync Server to communicate with external servers. The entry is important in that it contains the credentials (DN and password) used by the PingDataSync Server to access the source and target servers. The Sync User account resides in different entries depending on the targeted system:

- For the Ping Identity PingDirectory Server, Ping Identity PingDirectoryProxy Server, Nokia 8661 Directory Server, Nokia 8661 Directory Proxy Server, the Sync User account resides in the configuration entry (cn=Sync User, cn=Root DNs,cn=config).
- For Sun Directory Server, Sun DSEE, OpenDJ, Oracle Unified Directory, and generic LDAP directory topologies, the Sync User account resides under the base DN in the userRoot backend (cn=Sync User,dc=example,dc=com). The Sync User account should not reside in the cn=config branch for Sun Directory Server and DSEE machines.
- For Microsoft Active Directory servers, the Sync User account resides in the Users container (cn=Sync User,cn=Users,DC=adsync,DC=unboundid,DC=com).
- For Oracle and Microsoft SQL Servers, the Sync User account is a login account (SyncUser) with the sufficient privileges to access the tables to be synchronized.

In most cases, modifications to this account are rare. Make sure that the entry is not synchronized by setting up an optional Sync Class if the account resides in the userRoot backend (Sun Directory Server or Sun DSEE) or Users container (Microsoft Active Directory). For example, a Sync Class can be configured to have all CREATE, MODIFY, and DELETE operations set to false.

Configure the PingDataSync Server in Standard mode

The create-sync-pipe-config tool is used to configure Sync Pipes and Sync Classes. For bidirectional deployments, configure two Sync Pipes, one for each directional path.

Using the create-sync-pipe Tool to Configure Synchronization illustrates a bidirectional synchronization deployment in standard mode. The example assumes that two replicated topologies are configured:

- The first endpoint topology consists of two Sun LDAP servers: the main server and one failover. Both servers have Retro change logs enabled and contain the full DIT that will be synchronized to the second endpoint.
- The second endpoint topology consists of two PingDirectory Servers: the main server and one failover. Both servers have change logs enabled and contain entries similar to the first endpoint servers, except that they use a `mail` attribute, instead of an `email` attribute.

A specific `mail` to `email` mapping must also be created to exclude the source attribute on the Sync Pipe going the other direction.

Note
If the source attribute is not excluded, the PingDataSync Server will attempt to create an `email` attribute on the second endpoint, which could fail if the attribute is not present in the destination server's schema.

Then, two Sync Classes are defined:

- One to handle the customized `email` to `mail` attribute mapping.
- Another to handle all other cases (the default Sync Class).

The `dsconfig` command is used to create the specific attribute mappings. The `resync` command is used to test the mappings. Synchronization can start using the `realtime-sync` command.

Use the create-sync-pipe tool to configure synchronization

Use the `create-sync-pipe-config` utility to configure a Sync Pipe. Once the configuration is completed, settings can be adjusted using the `dsconfig` tool.

Note
If servers have no base entries or data, the `cn=Sync User,cn=Root DNs,cn=config` account needed to communicate cannot be created. Make sure that base entries are created on the destination servers.

If synchronizing pre-encoded passwords to a Ping PingDirectory Server destination, allow pre-encoded passwords in the default password policy. Password encryption must also be configured on the destination. Be sure that the password encryption algorithm is supported by both source and destination servers with the following command:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set allow-pre-encoded-passwords:true
```

Encrypted and clear-text passwords can be synchronized by configuring the Sync Destination `password-synchronization-format`, and `require-secure-connection-for-clear-text-passwords` properties.

Note
The `require-secure-connection-for-clear-text-passwords` property can be set to `false` when working in a test environment. If the `password-synchronization-format` property is set to `clear-text`, and `require-secure-connection-for-clear-text-passwords` property is set

to `true`, the connection must be secure. If a secure connection is not available, an error is generated and the password is not synchronized.

Perform the following steps to configure the PingDataSync Server using `create-sync-pipe-config`:

1. Start the PingDataSync Server.

```
$ <server-root>/bin/start-server
```

2. From the `bin` directory, run the `create-sync-pipe-config` tool.

```
$ bin/create-sync-pipe-config
```

3. On the Initial Synchronization Configuration Tool menu, press **Enter** (yes) to continue the configuration.
4. On the Synchronization Mode menu, press **Enter** to select Standard Mode.
5. On the Synchronization Directory menu, select **oneway (1)** or **bidirectional (2)** for the synchronization topology. This example assumes bidirectional synchronization.
6. On the Source Endpoint Type menu, select the directory or database server for the first endpoint.
7. On the Source Endpoint Name menu, type a name for the endpoint server, or press **Enter** accept the default.
8. On the Base DNs menu, type the base DN on the first endpoint topology where the entries will be searched. In this example, (`dc=example,dc=com`) is used.
9. Select an option for the server security.
10. Type the host name and listener port number for the source server, or accept the default. Make sure that the endpoint servers are online and running.
11. Enter another server host and port, or press **Enter** to continue.
12. Enter the Sync User account DN for the endpoint servers, or press **Enter** to accept the default (`cn=Sync User,cn=Root DNs,cn=config`).
13. Enter and confirm a password for this account.
14. The servers in the destination endpoint topology can now be configured. Repeat steps 6–13 to configure the second server.
15. Define the maximum age of changelog log entries, or press **Enter** to accept the default.
16. After the source and destination topologies are configured, the PingDataSync Server will "prepare" each external server by testing the connection to each server. This step determines if each account has the necessary privileges (root privileges are required) to communicate with and transfer data to each endpoint during synchronization.
17. Create a name for the Sync Pipe on the Sync Pipe Name menu, or press **Enter** to accept the default. Because this configuration is bidirectional, the following step is setting up a

Sync Pipe path from the source endpoint to the destination endpoint. A later step will define another Sync Pipe from the PingDirectory Server to another server.

18. On the Sync Class Definitions menu, type **Yes** to create a custom Sync Class. A Sync Class defines the operation types (creates, modifies, or deletes), attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated.
19. Enter a name for the new Sync Class, such as "server1_to_server2."
20. On the Base DNs for Sync Class menu, enter one or more base DNs to synchronize specific subtrees of a DIT. Entries outside of the specified base DNs are excluded from synchronization. Make sure the base DNs do not overlap.
21. On the Filters for Sync Class menu, define one or more LDAP search filters to restrict specific entries for synchronization, or press **Enter** to accept the default (no). Entries that do not match the filters will be excluded from synchronization.
22. On the Synchronized Attributes for Sync Class menu, specify which attributes will be automatically mapped from one system to another. This example will exclude the source attribute (`email`) from being auto-mapped to the target servers.
23. On the Operations for Sync Class menu, select the operations that will be synchronized for the Sync Class, or press **Enter** to accept the default (`1, 2, 3`).
24. Define a default Sync Class that specifies how the other entries are processed, or press **Enter** to create a Sync Class called "Default Sync Class."
25. On the Default Sync Class Operations menu, specify the operations that the default Sync Class will handle during synchronization, or press **Enter** to accept the default.
26. Define a Sync Pipe going from the PingDirectory Server to the Sun Directory Server and exclude the `mail` attribute from being synchronized to the other endpoint servers.
27. Review the Sync Pipe Configuration Summary, and press **Enter** to accept the default (write configuration), which records the commands in a batch file (`<server-root>/sync-pipe-cfg.txt`). The batch file can be re-used to set up other topologies.

Apply the configuration changes to the local PingDataSync Server instance using a `dsconfig` batch file. Any Server SDK extensions, should be saved to the `<server-root>/lib/extensions` directory.

The next step will be to configure the attribute mappings using the `dsconfig` command.

Configuring attribute mapping

The following procedure defines an attribute map from the `email` attribute in the source servers to a `mail` attribute in the target servers. Both attributes must be valid in the target servers and must be present in their respective schemas.

Note
The following can also be done with `dsconfig` in interactive mode. Attribute mapping options are available from the PingDataSync Server main menu.

1. On the PingDataSync Server, run the `dsconfig` command to create an attribute map for the "SunDS>DS" Sync Class for the "Sun DS to Ping Identity DS" Sync Pipe, and then run the second `dsconfig` command to apply the new attribute map to the Sync Pipe and Sync Class.

```
$ bin/dsconfig --no-prompt create-attribute-map \
  --map-name "SunDS>DS Attr Map" \
  --set "description:Attribute Map for SunDS>Ping Identity Sync Class" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name "Sun DS to DS" \
  --class-name "SunDS>DS" \
  --set "attribute-map:SunDS>DS Attr Map" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. Create an attribute mapping (from `email` to `mail`) for the new attribute map.

```
$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name "SunDS>DS Attr Map" \
  --mapping-name mail --type direct \
  --set "description:Email>Mail Mapping" \
  --set from-attribute:email \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

3. For a bidirectional deployment, repeat steps 1–2 to create an attribute map for the DS>SunDS Sync Class for the Ping Identity DS to Sun DS Sync Pipe, and create an attribute mapping that maps `mail` to `email`.

```
$ bin/dsconfig --no-prompt create-attribute-map \
  --map-name "DS>SunDS Attr Map" \
  --set "description:Attribute Map for DS>SunDS Sync Class" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name "Ping Identity DS to Sun DS" \
  --class-name "DS>SunDS" \
  --set "attribute-map:DS>SunDS Attr Map" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name "DS>SunDS Attr Map" \
  --mapping-name email \
  --type direct \
  --set "description:Mail>Email Mapping" \
  --set from-attribute:mail \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configure server locations

The PingDataSync Server supports endpoint failover, which is configurable using the `location` property on the external servers. By default, the server prefers to connect to, and failover to, endpoints in the same location as itself. If there are no location settings configured, the PingDataSync Server will iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

> **Note**
> Location-based failover is only applicable for LDAP endpoint servers.

1. On the PingDataSync Server, run the `dsconfig` command to set the location for each external server in the Sync Source and Sync Destination. For example, the following command sets the location for six servers in two data centers, `austin` and `dallas`.

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:1389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:2389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:3389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:4389 \
  --set location:dallas
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:5389 \
  --set location:dallas
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:6389 \
  --set location:dallas
```

2. Run `dsconfig` to set the location on the Global Configuration. This is the location of the PingDataSync Server itself. In this example, set the location to "austin."

```
$ bin/dsconfig set-global-configuration-prop \
  --set location:austin
```

Use the Configuration API

PingData servers provide a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers should allow the `application/json` content type.

The server includes a servlet extension that provides read and write access to the server's configuration over HTTP. The extension is enabled by default for new installations, and can be enabled for existing deployments by simply adding the extension to one of the server's HTTP Connection Handlers, as follows:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add http-servlet-extension:Configuration
```

The API is made available on the HTTPS Connection handler's `host:port` in the `/config` context. Due to the potentially sensitive nature of the server's configuration, the HTTPS Connection Handler should be used, for hosting the Configuration extension.

Authentication and authorization

Clients must use HTTP Basic authentication to authenticate to the Configuration API. If the username value is not a DN, then it will be resolved to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper, or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. For example:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Configuration \
  --set "identity-mapper:Alternative Identity Mapper”
```

To access configuration information, users must have the appropriate privileges:

- To access the `cn=config` backend, users must have the `bypass-acl` privilege or be allowed access to the configuration using an ACI.
- To read configuration information, users must have the `config-read` privilege.
- To update the configuration, users must have the `config-write` privilege.

Relationship between the Configuration API and the dsconfig tool

The Configuration API is designed to mirror the `dsconfig` tool, using the same names for properties and object types. Property names are presented as hyphen case in `dsconfig` and as camel-case attributes in the API. In API requests that specify property names, case is not important. Therefore, `baseDN` is the same as `baseDn`. Object types are represented in hyphen case. API paths mirror what is in dsconfig. For example, the `dsconfig list-connection-handlers` command is analogous to the API's `/config/connection-handlers` path. Object types that appear in the schema URNs adhere to a `type:subtype` syntax. For example, a Local

DB Backend's schema URN is urn:unboundid:schemas:configuration:2.0:backend:local-db. Like the dsconfig tool, all configuration updates made through the API are recorded in logs/config-audit.log.

The API includes the filter, sort, and pagination query parameters described by the SCIM specification. Specific attributes may be requested using the attributes query parameter, whose value must be a comma-delimited list of properties to be returned, for example attributes=baseDN,description. Likewise, attributes may be excluded from responses by specifying the excludedAttributes parameter. See Sorting and Filtering with the Configuration API for more information on query parameters.

Operations supported by the API are those typically found in REST APIs:

HTTP Method	Description	Related dsconfig Example
GET	Lists the attributes of an object when used with a path representing an object, such as /config/global-configuration or /config/backends/userRoot. Can also list objects when used with a path representing a parent relation, such as /config/backends.	get-backend-prop list-backends get-global-configuration-prop
POST	Creates a new instance of an object when used with a relation parent path, such as config/backends.	create-backend
PUT	Replaces the existing attributes of an object. A PUT operation is similar to a PATCH operation, except that the PATCH is determined by determining the difference between an existing target object and a supplied source object. Only those attributes in the source object are modified in the target object. The target object is specified using a path, such as /config/backends/userRoot.	set-backend-prop set-global-configuration-prop
PATCH	Updates the attributes of an existing object when used with a path representing an object, such as /config/backends/userRoot. See PATCH Example.	set-backend-prop set-global-configuration-prop
DELETE	Deletes an existing object when used with a path representing an object, such as /config/backends/userRoot.	delete-backend

The OPTIONS method can also be used to determine the operations permitted for a particular path.

Object names, such as userRoot in the Description column, must be URL-encoded in the path segment of a URL. For example, %20 must be used in place of spaces, and %25 is used in place of the percent (%) character. So the URL for accessing the HTTP Connection Handler object is:

```
/config/connection-handlers/http%20connection%20handler
```

GET Example

The following is a sample GET request for information about the userRoot backend:

```
GET /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
```

The response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://localhost:5033/config/backends/userRoot"
  },
  "backendID": "userRoot2",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example2,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "10",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "1 m",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "0",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "0",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode": "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "enabled": "false",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "false",
  "id2childrenIndexEntryLimit": "66",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "jeProperty": [
    "je.cleaner.adjustUtilization=false",
    "je.nodeMaxEntries=32"
  ],
  "numRecentChanges": "50000",
  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "5000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

GET list example

The following is a sample GET request for all local backends:

```
GET /config/backends
Host: example.com:5033
Accept: application/scim+json
```

The response (which has been shortened):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=admin data"
      ],
      "enabled": "true",
      "isPrivateBackend": "true",
      "javaClass": "com.unboundid.directory.server.backends.LDIFBackend",
      "ldifFile": "config/admin-backend.ldif",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "false",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
      ],
      "id": "ads-truststore",
      "meta": {
        "resourceType": "Trust Store Backend",
        "location": "http://localhost:5033/config/backends/ads-truststore"
      },
      "backendID": "ads-truststore",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=ads-truststore"
      ],
      "enabled": "true",
      "javaClass":
"com.unboundid.directory.server.backends.TrustStoreBackend",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "true",
      "trustStoreFile": "config/server.keystore",
      "trustStorePin": "********",
      "trustStoreType": "JKS",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:alarm"
      ],
      "id": "alarms",
      "meta": {
        "resourceType": "Alarm Backend",
        "location": "http://localhost:5033/config/backends/alarms"
      },
 ...
```

PATCH example

Configuration can be modified using the HTTP PATCH method. The PATCH request body is a JSON object formatted according to the SCIM patch request. The Configuration API, supports a subset of possible values for the `path` attribute, used to indicate the configuration attribute to modify.

The configuration object's attributes can be modified in the following ways. These operations are analogous to the `dsconfig modify-[object]` options.

- An operation to set the single-valued `description` attribute to a new value:

  ```
  {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --set "description:A new backend"
  ```

- An operation to add a new value to the multi-valued `jeProperty` attribute:

  ```
  {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --add je-property:je.env.backgroundReadLimit=0
  ```

- An operation to remove a value from a multi-valued property. In this case, `path` specifies a SCIM filter identifying the value to remove:

  ```
  {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --remove je-property:je.cleaner.adjustUtilization=false
  ```

- A second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify, and a SCIM filter whose attribute is `value`:

  ```
  {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }
  ```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --remove je-property:je.nodeMaxEntries=32
```

- An option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value:

```
{
  "op" : "remove",
  "path" : "id2childrenIndexEntryLimit"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --reset id2childrenIndexEntryLimit
```

The following is the full example request. The API responds with the entire modified configuration object, which may include a SCIM extension attribute `urn:unboundid:schemas:configuration:messages` containing additional instructions:

Example request:

```
PATCH /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json

{
  "schemas" : [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  } ]
}
```

Example response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot2",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot2"
  },
  "backendID": "userRoot2",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example2,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "10",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "1 m",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "0",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "0",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode": "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "123",
  "enabled": "false",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "false",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "jeProperty": [
  "\"je.env.backgroundReadLimit=0\""
  ],
  "numRecentChanges": "50000",
  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "5000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled",
  "urn:unboundid:schemas:configuration:messages:2.0": {
    "requiredActions": [
      {
        "property": "jeProperty",
        "type": "componentRestart",
        "synopsis": "In order for this modification to take effect,
         the component must be restarted, either by disabling and
         re-enabling it, or by restarting the server"
      },
      {
        "property": "id2childrenIndexEntryLimit",
        "type": "other",
        "synopsis": "If this limit is increased, then the contents
         of the backend must be exported to LDIF and re-imported to
         allow the new limit to be used for any id2children keys
         that had already hit the previous limit."
      }
    ]
  }
}
```

API paths

The Configuration API is available under the `/config` path. A full listing of root sub-paths can be obtained from the `/config/ResourceTypes` endpoint:

```
GET /config/ResourceTypes
Host: example.com:5033
Accept: application/scim+json
```

Sample response (abbreviated):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "dsee-compat-access-control-handler",
      "name": "DSEE Compat Access Control Handler",
      "description": "The DSEE Compat Access Control
         Handler provides an implementation that uses syntax
         compatible with the Sun Java System Directory Server
         Enterprise Edition access control handler.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/dsee-compat-
access-control-handler"
      }
    },
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "access-control-handler",
      "name": "Access Control Handler",
      "description": "Access Control Handlers manage the
         application-wide access control. The server's access
         control handler is defined through an extensible
         interface, so that alternate implementations can be created.
         Only one access control handler may be active in the server
         at any given time.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/access-
control-handler"
```

```
      }
    },
    {
...
```

The response's `endpoint` elements enumerate all available sub-paths. The path `/config/access-control-handler` in the example can be used to get a list of existing access control handlers, and create new ones. A path containing an object name like `/config/backends/{backendName}`, where `{backendName}` corresponds to an existing backend (such as `userRoot`) can be used to obtain an object's properties, update the properties, or delete the object.

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted nor created. These paths can be differentiated from others by their singular, rather than plural, relation name (for example `global-configuration`).

Sorting and filtering configuration objects

The Configuration API supports SCIM parameters for filter, sorting, and pagination. Search operations can specify a SCIM filter used to narrow the number of elements returned. See the SCIM specification for the full set of operations for SCIM filters. Clients may also specify sort parameters, or paging parameters. As previously mentioned, clients may specify attributes to include or exclude in both get and list operations.

GET Parameters for Sorting and Filtering

GET Parameter	Description
filter	Values can be simple SCIM filters such as `id eq "userRoot"` or compound filters like `meta.resourceType eq "Local DB Backend"` and `baseDn co "dc=exmple,dc=com"`.
sortBy	Specifies a property value by which to sort.
sortOrder	Specifies either `ascending` or `descending` alphabetical order.
startIndex	1-based index of the first result to return.
count	Indicates the number of results per page.

Update properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH. With PUT, the server computes the differences between the object in the request with the current version in the server, and performs modifications where necessary. The server will never remove attributes that are not specified in the request. The API responds with the entire modified object.

Request:

```
PUT /config/backends/userRoot
Host: example.com:5033
```

```
Accept: application/scim+json
{
  "description" : "A new description."
}
```

Response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode": "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "abc",
  "enabled": "true",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "true",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "numRecentChanges": "50000",
  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative actions

Updating a property may require an administrative action before the change can take effect. If so, the server will return `200 Success`, and any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

For example, changing the `jeProperty` of a backend will result in the following:

```
"urn:unboundid:schemas:configuration:messages:2.0": {
  "requiredActions": [
    {
      "property": "baseContextPath",
      "type": "componentRestart",
      "synopsis": "In order for this modification to
         take effect, the component must be restarted,
         either by disabling and re-enabling it, or by
         restarting the server"
    },
    {
      "property": "id2childrenIndexEntryLimit",
      "type": "other",
      "synopsis": "If this limit is increased, then the
         contents of the backend must be exported to LDIF
         and re-imported to allow the new limit to be used
         for any id2children keys that had already hit the
         previous limit."
    }
  ]
}
...
```

Update servers and server groups

Servers can be configured as part of a server group, so that configuration changes that are applied to a single server, are then applied to all servers in a group. When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query parameter. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `singleServer` or `serverGroup` can be specified. For example:

```
https://example.com:5033/config/Backends/userRoot?applyChangeTo=singleServer
```

Note
This does not apply to mirrored subtree objects, which include Topology and Cluster level objects. Changes made to mirrored objects are applied to all objects in the subtree.

Configuration API Responses

Clients of the API should examine the HTTP response code in order to determine the success or failure of a request. The following are response codes and their meanings:

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, or object properties, administrative actions.
204 No Content	The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation.	None.
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional

Response Code	Description	Response Body
		message.
401 Unauthorized	User authentication is required. Some user agents such as browsers may respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid.	None.
403 Forbidden	The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted.	None.
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message.
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists or an attempt was made to delete an object that is referred to by another object.	Error summary and optional message.
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None.
500 Server Error	The server encountered an unexpected error. Please report server errors to customer support.	Error summary and optional message.

An application that uses the Configuration API should limit dependencies on particular text appearing in error message content. These messages may change, and their presence may depend on server configuration. Use the HTTP return code and the context of the request to create a client error message. The following is an example encoded error message:

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "status": 404,
  "scimType": null,
  "detail": "The Local DB Index does not exist."
}
```

Configuration with the dsconfig tool

The Ping Identity servers provide several command-line tools for management and administration. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

The `dsconfig` tool is the text-based management tool used to configure the underlying server configuration. The tool has three operational modes:

- Interactive mode
- Non-interactive mode

- Batch mode

The `dsconfig` tool also offers an offline mode using the `--offline` option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. View detailed argument options and examples by typing `--help` with the command.

```
$ bin/dsconfig --help
```

To list the subcommands for each command:

```
$ bin/dsconfig --help-subcommands
```

To list more detailed subcommand information:

```
$ bin/dsconfig list-log-publishers --help
```

Use dsconfig in interactive mode

Running `dsconfig` in interactive command-line mode provides a menu-driven interface for accessing and configuring the PingData server. To start `dsconfig` in interactive mode, run the tool without any arguments:

```
$ bin/dsconfig
```

Running the tool requires server connection and authentication information. After connection information is confirmed, a menu of the available operation types is displayed.

Use dsconfig in non-interactive mode

Non-interactive command-line mode provides a simple way to make arbitrary changes to the server, and to use administrative scripts to automate configuration changes. To make changes to multiple configuration objects at the same time, use batch mode.

The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The `--no-prompt` argument specifies non-interactive mode. The `{globalArgs}` argument provides a set of arguments that specify how to connect and authenticate to the server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on the implementation. The `{subcommand}` specifies which general action to perform. The following uses standard LDAP connections:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

The following uses SASL GSSAPI (Kerberos) parameters:

```
$ bin/dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The `{subcommandArgs}` argument contains a set of arguments specific to the particular task. To always display the advanced properties, use the `--advanced` command-line option.

> **Note**
> Global arguments can appear anywhere on the command line. The subcommand-specific arguments can appear anywhere after the subcommand.

Use dsconfig batch mode

The `dsconfig` tool provides a batching mechanism that reads multiple invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The `--no-prompt` option is required with `dsconfig` in batch mode.

```
$ bin/dsconfig --no-prompt \
  --hostname host1 \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --batch-file /path/to/sync-pipe-config.txt
```

If a `dsconfig` command has a missing or incorrect argument, the command will fail and stop the batch process without applying any changes to the server. A `--batch-continue-on-error` option is available, which instructs `dsconfig` to apply all changes and skip any errors.

View the `logs/config-audit.log` file to review the configuration changes made to the server, and use them in the batch file. The batch file can have blank lines for spacing, and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The PingDataSync Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to Ping Identity PingDataSync Server machines.

Topology configuration

Topology configuration enables grouping servers and mirroring configuration changes automatically. It uses a master/slave architecture for mirroring shared data across the topology. All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group.

> **Note**
> To remove a server from the topology, it must be uninstalled with the `uninstall` tool.

Topology master requirements and selection

A topology master server receives any configuration change from other servers in the topology, verifies the change, then makes the change available to all connected servers when they poll the master. The master always sends a digest of its subtree contents on each update. If the node has a different digest than the master, it knows it's not synchronized. The servers will pull the entire subtree from the master if they detect that they are not synchronized. A server may detect it is not synchronized with the master under the following conditions:

- At the end of its periodic polling interval, if a server's subtree digest differs from that of its master, then it knows it's not synchronized.
- If one or more servers have been added to or removed from the topology, the servers will not synchronized.

The master of the topology is selected by prioritizing servers by minimum supported product version, most available, newest server version, earliest start time, and startup UUID (a smaller UUID is preferred).

After determining a master, the topology data is reviewed from all available servers (every five seconds by default) to determine if any new information makes a server better suited to being the master. If a new server can be the master, it will communicate that to the other servers, if no other server has advertised that it should be the master. This ensures that all servers accept the same master at approximately the same time (within a few milliseconds of each other). If there is no better master, the initial master maintains the role.

After the best master has been selected for the given interval, the following conditions are confirmed:

- A majority of servers is reachable from that master. (The master server itself is considered while determining this majority.)
- There is only a single master in the entire topology.

If either of these conditions is not met, the topology is without a master and the peer polling frequency is reduced to 100 milliseconds to find a new master as quickly as possible. If there is no master in the topology for more than one minute, a `mirrored-subtree-manager-no-master-found` alarm is raised. If one of the servers in the topology is forced as master with the `force-as-master-for-mirrored-data` option in the Global Configuration configuration object, a `mirrored-subtree-manager-forced-as-master-warning` warning alarm is raised. If multiple servers have been forced as masters, then a `mirrored-subtree-manager-forced-as-master-error` critical alarm will be raised.

Topology components

When a server is installed, it can be added to an existing topology, which will clone the server's . Topology settings are designed to operate without additional configuration. If required, some settings can be adjusted to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the Global Configuration and in the Config File Handler Backend. Though they are topology settings, they are unique to each server and are not mirrored. Settings must be kept the same on all servers.

The Global Configuration object contains a single topology setting, `force-as-master-for-mirrored-data`. This should be set to `true` on only one of the servers in the topology, and is used only if a situation occurs where the topology cannot determine a master because a majority of servers is not available. A server with this setting enabled will be assigned the role of master, if no suitable master can be determined. See Topology master requirements and selection for details about how a master is selected for a topology.

The Config File Handler Backend defines three topology (`mirrored-subtree`) settings:

- `mirrored-subtree-peer-polling-interval` – Specifies the frequency at which the server polls its topology peers to determine if there are any changes that may warrant a new master selection. A lower value will ensure a faster failover, but it will also cause more traffic among the peers. The default value is five seconds. If no suitable master is found, the polling frequency is adjusted to 100 milliseconds until a new master is selected.
- `mirrored-subtree-entry-update-timeout` – Specifies the maximum length of time to wait for an update operation (add, delete, modify or modify-dn) on an entry to be applied by the master on all of the servers in the topology. The default is 10 seconds. In reality, updates can take up to twice as much time as this timeout value if master selection is in progress at the time the update operation was received.
- `mirrored-subtree-search-timeout` – Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

Topology meta-data is stored under the `cn=topology,cn=config` subtree and cluster data is stored under the `cn=cluster,cn=config` subtree. The only setting that can be changed is the cluster name.

Monitor data for the topology

Each server has a monitor that exposes that server's view of the topology in its monitor backend, so that peer servers can periodically read this information to determine if there are changes in the topology. Topology data includes the following:

- The server ID of the current master, if the master is not known.
- The instance name of the current master, or if a master is not set, a description stating why a master is not set.
- A flag indicating if this server thinks that it should be the master.

- A flag indicating if this server is the current master.
- A flag indicating if this server was forced as master.
- The total number of configured peers in the topology group.
- The peers connected to this server.
- The current availability of this server
- A flag indicating whether or not this server is not synchronized with its master, or another node in the topology if the master is unknown.
- The amount of time in milliseconds where multiple masters were detected by this server.
- The amount of time in milliseconds where no suitable server is found to act as master.
- A SHA-256 digest encoded as a base-64 string for the current subtree contents.

The following metrics are included if this server has processed any operations as master:

- The number of operations processed by this server as master.
- The number of operations processed by this server as master that were successful.
- The number of operations processed by this server as master that failed to validate.
- The number of operations processed by this server as master that failed to apply.
- The average amount of time taken (in milliseconds) by this server to process operations as the master.
- The maximum amount of time taken (in milliseconds) by this server to process an operation as the master.

Updating the server instance listener certificate

To change the SSL certificate for the server, update the keystore and truststore files with the new certificate. The certificate file must have the new certificate in PEM-encoded format, such as:

```
-----BEGIN CERTIFICATE-----

MIIDKTCCAhGgAwIBAgIEacgGrDANBgkqhkiG9w0BAQsFADBFMR4wHAYDVQQKExVVbmJvdW5kSUQgQ2
VydGlmaWNhdGUxIzAhBgNVBAMTGnZtLW1lZGl1bS03My51bmJvdW5kaWQubGFiMB4XDTE1MTAxMjE1
MzU0OFoXDTM1MTAwNzE1MzU0OFowRTEeMBwGA1UEChMVVW5ib3VuZElEIENlcnRpZmljYXRlMSMwIQ
YDVQQDExp2bS1tZWRpdW0tNzMudW5ib3VuZGlkLmxhYjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBAKN4tAN3o9Yw6Cr9hivwVDxJqF6+aEi9Ir3WGFYLSrggRNXsiAOfWkSMWdIC5vyF5OJ9Dl
IgvHL4OuqP/YNEGzKDkgr6MwtUeVSK14+dCixygJGC0nY7k+f0WSCjtIHzrmc4WWdrZXmgb+qv9Lup
S30JG0FXtcbGkYpjaKXIEqMg4ekz3B5cAvE0SQUFyXEdN4rWOn96nVFkb2CstbiPzAgne2tu7paJ6S
GFOW0UF7v018XY1m2WHBIoD0WC8nOVLTG9zFUavaOxtlt1TlhClkI4HRMNg8n2EtSTdQRizKuw9DdT
XJBb6Kfvnp/nI73VHRyt47wUVueehEDfLtDP8pMCAwEAAaMhMB8wHQYDVR0OBBYEFMrwjWxl2K+yd9
+Y65oKn0g5jITgMA0GCSqGSIb3DQEBCwUAA4IBAQBpsBYodblUGew+HewqtO2i8Wt+vAbt31zM5/kR
vo6/+iPEASTvZdCzIBcgletxKGKeCQ0GPeHr42+erakiwmGDlUTYrU3LU5pTGTDLuR2IllTT5xlEhC
```

```
WJGWipW4q3Pl3cX/9m2ffY/JLYDfTJaoJvnXrh7Sg719skkHjWZQgOHXlkPLx5TxFGhAovE1D4qLVR
WGohdpWDrIgFh0DVfoyAn1Ws9ICCXdRayajFI4Lc6K1m6SA5+25Y9nno8BhVPf4q5OW6+UDc8MsLbB
sxpwvR6RJ5cv3ypfOriTehJsG+9ZDo7YeqVsTVGwAlW3PiSd9bYP/8yu9Cy+0MfcWcSeAE
-----END CERTIFICATE-----
```

If clients that already have a secure connection established with this server need to be maintained, information about both certificates can reside in the same file (each with their own begin and end headers and footers).

After the keystore and truststore files are updated, run the following `dsconfig` command to update the server's certificate in the topology registry:

```
$ bin/dsconfig set-server-instance-listener-prop \
  --instance-name <server-instance-name> \
  --listener-name ldap-listener-mirrored-config \
  --set listener-certificate <path-to-new-certificate-file>
```

The `listener-certificate` in the topology registry is like a trust store. The public certificates that it has are automatically trusted by the local server. When the local server attempts a secure LDAP connection to a peer, and the peer presents it with its certificate, the local server will check the `listener-certificate` property for that server in the topology registry. If the property contains the peer server's certificate, the local server will trust the peer.

Remove the self-signed certificate

The server is installed with a self-signed certificate and key (`ads-certificate`), which are used for internal purposes such as replication authentication, inter-server authentication in the topology registry, reversible password encryption, and encrypted backup/LDIF export. The ads-certificate lives in the keystore file called `ads-truststore` under the server’s `/config` directory. If your deployment requires removing the self-signed certificate, it can be replaced.

The certificate is stored in the topology registry, which enables replacing it on one server and having it mirrored to all other servers in the topology. Any change is automatically mirrored on other servers in the topology. It is stored in human-readable PEM-encoded format and can be updated with `dsconfig`. The following general steps are required to replace the self-signed certificate:

1. Prepare a new keystore with the replacement key-pair.
2. Update the server configuration to use the new certificate by adding it to the server’s list of certificates in the topology registry so that it is trusted by other servers.
3. Update the server’s `ads-truststore` file to use the new key-pair.
4. Retire the old certificate by removing it from the topology registry.

Note

Replacing the entire key-pair instead of just the certificate associated with the original private key can make existing backups and LDIF exports invalid. This should be performed immediately after setup or before the key-pair is used. After the first time, only the certificate associated with the private key should have to be changed, for example, to extend its validity period or replace it with a certificate signed by a different CA.

Prepare a new keystore with the replacement key-pair

The self-signed certificate can be replaced with an existing key-pair, or the certificate associated with the original key-pair can be used.

Use an existing key-pair

If a private key and certificate(s) in PEM-encoded format already exist, both the original private key and self-signed certificate can be replaced in ads-truststore with the `manage-certificates` tool. The following command imports existing certificates into a new keystore file, `ads-truststore.new`:

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file existing.key \
  --certificate-file existing.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

The certificates listed using the `--certificate-file` options must be ordered so that each subsequent certificate is the issuer for the previous one. So the server certificate comes first, the intermediate certificates next (if any), and the root CA certificate last.

Use the certificate associated with the original key-pair

The certificate associated with the original server-generated private key can be replaced with the following commands:

1. Create a CSR for the `ads-certificate`:

```
$ bin/manage-certificates generate-certificate-signing-request \
  --keystore ads-truststore \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --use-existing-key-pair \
  --subject-dn "CN=ldap.example.com,O=Example Corporation,C=US" \
  --output-file ads.csr
```

2. Submit `ads.csr` to a CA for signing.

3. Export the server’s private key into `ads.key`:

```
$ bin/manage-certificates export-private-key \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --output-file ads.key
```

4. Import the certificates obtained from the CA (the CA-signed server certificate, any intermediate certificates, and root CA certificate) into `ads-truststore.new`:

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file ads.key \
  --certificate-file new-ads.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

Update the server configuration to use the new certificate

To update the server to use the desired key-pair, the `inter-server-certificate` property for the server instance must first be updated in the topology registry. The old and the new certificates may appear within their own begin and end headers in the `inter-server-certificate` property to support transitioning from the old certificate to the new one.

1. Export the server's old ads-certificate into `old-ads.crt`:

```
$ bin/manage-certificates export-certificate \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --export-certificate-chain \
  --output-file old-ads.crt
```

2. Concatenate the old, new certificate, and issuer certificates into one file. On Windows, an editor like notepad can be used. On Unix platforms, use the following command:

```
$ cat old-ads.crt new-ads.crt intermediate.crt root-ca.crt > chain.crt
```

3. Update the `inter-server-certificate` property for the server instance in the topology registry using `dsconfig`:

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name <instance-name> \
  --set “inter-server-certificate<chain.crt”
```

Update the ads-truststore file to use the new key-pair

The server will still use the old `ads-certificate`. When the new `ads-certificate` needs to go into effect, the old `ads-truststore` file must be replaced with `ads-truststore.new` in the server's `config` directory.

```
$ mv ads-truststore.new ads-truststore
```

Retire the old certificate

The old certificate is retired by removing it from the topology registry when it has expired. All existing encrypted backups and LDIF exports are not affected because the public key in the old and new server certificates are the same, and the private key will be able to decrypt them.

```
$ cat new-ads.crt intermediate.crt root-ca.crt > chain.crt
```

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name <instance-name> \
  --set “inter-server-certificate<chain.crt”
```

Domain Name Service (DNS) caching

If needed, two global configuration properties can be used to control the caching of hostname-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system. Use the `dsconfig` tool to configure these properties.

network-address-cache-ttl– Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a hostname-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.

network-address-outage-cache-enabled – Caches hostname-to-IP address results in the event of a DNS outage. This is set to `true` by default, meaning name resolution results are cached. Unexpected service interruptions may occur during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

IP address reverse name lookups

Ping Identity servers do not explicitly perform numeric IP address-to-hostname lookups. However, address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing may trigger implicit reverse name lookups. For more information about how address masks are configured in the server, review the following information for each server:

- ACI dns: bind rules under *Managing Access Control* (PingDirectory Server and PingDirectoryProxy Servers)
- `ds-auth-allowed-address`: *Adding Operational Attributes that Restrict Authentication* (PingDirectory Server)
- Connection Criteria: *Restricting Server Access Based on Client IP Address* (PingDirectory Server and PingDirectoryProxy Servers)
- Connection Handlers: restrict server access using Connection Handlers (Configuration Reference Guide for all PingData servers)

Configure the synchronization environment with dsconfig

The dsconfig tool can be used to configure any part of the PingDataSync Server, but will likely be used for more fine-grained adjustments. If configuring a Sync Pipe for the first time, use the bin/create-sync-pipe-config tool to guide through the necessary Sync Pipes creation steps.

Configure server groups with dsconfig interactive

In a typical deployment, onePingDataSync Server and one or more redundant failover servers are configured. Primary and secondary servers must have the same configuration settings to ensure proper operation. To enable this, assign all servers to a server group using the dsconfig tool. Any change to one server will automatically be applied to the other servers in the group.

Run the dsconfig command and set the global configuration property for server groups to all-servers. On the primary PingDataSync Server, do the following:

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:all-servers
```

Updates to servers in the group are made using the --applyChangeTo servergroup option of the dsconfig command. To apply the change to one server in the group, use the --applyChangeTo single-server option. If additional servers are added to the topology, the setup tool will copy the configuration from the primary server to the new server(s).

Start the Global Sync Configuration with dsconfig interactive

After the Synchronization topology is configured, perform the following steps to start the Global Sync Configuration, which will use only those Sync Pipes that have been started:

1. On the dsconfig main menu, type the number corresponding to the Global Sync Configuration.

2. On the Global Sync Configuration menu, type the number corresponding to view and edit the configuration.

3. On the Global Sync Configuration Properties menu, type the number corresponding to setting the started property, and then follow the prompts to set the value to TRUE.

4. On the Global Sync Configuration Properties menu, type **f** to save and apply the changes.

Prepare external server communication

The prepare-endpoint-server tool sets up any communication variances that may occur between the PingDataSync Server and the external servers. Typically, directory servers can have different security settings, privileges, and passwords configured on the Sync Source that might reject the import of entries in the Sync Destination.

The prepare-endpoint-server tool also creates a Sync User Account and its privileges on all of the external servers (see About the Sync User Account for more detailed information). The prepare-endpoint-server tool verifies that the account has the proper privileges to access the firstChangeNumber and lastChangeNumber attributes in the root DSE entry so that it can access the latest changes. If the Sync User does not have the proper privileges, the PingDataSync Server displays a warning message, which is saved in the logs/prepare-endpoint-server.log file.

Note
If the synchronization topology was created using the create-sync-pipe-config tool, this command does not need to be run. It is already part of the create-sync-pipe-config process.

Perform the following steps to prepare thePingDataSync Server for external server communication:

1. Use the prepare-endpoint-server tool to prepare the directory server instances on the remote host for synchronization as a data source for the subtree, dc=example,dc=com. If the user account is not present on the external server, it will be created if a parent entry exists.

```
$ bin/prepare-endpoint-server \
  --hostname sun-ds1.example.com \
  --port 21389 \
  --syncServerBindDN "cn=Sync User,dc=example,dc=com" \
  --syncServerBindPassword secret \
  --baseDN "dc=example,dc=com" \
  --isSource
```

2. When prompted, enter the bind DN and password to create the user account. This step enables the change log database and sets the changelog-maximum-age property.
3. Repeat steps 1–2 for any other external source servers.
4. For the destination servers, repeat steps 2–3 and include the --isDestination option. If destination servers do not have any entries, a "Denied" message will display when creating the cn=Sync User entry.

```
$ bin/prepare-endpoint-server \
  --hostname PingIdentity-ds1.example.com \
  --port 33389 \
  --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
  --syncServerBindPassword sync \
  --baseDN "dc=example,dc=com" \
  --isDestination
```

5. Repeat step 4 for any other destination servers.

Configuration with the dsconfig tool

The Ping Identity servers provide several command-line tools for management and administration. The command-line tools are available in the bin directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

The `dsconfig` tool is the text-based management tool used to configure the underlying server configuration. The tool has three operational modes:

- Interactive mode
- Non-interactive mode
- Batch mode

The `dsconfig` tool also offers an offline mode using the `--offline` option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. View detailed argument options and examples by typing `--help` with the command.

```
$ bin/dsconfig --help
```

To list the subcommands for each command:

```
$ bin/dsconfig --help-subcommands
```

To list more detailed subcommand information:

```
$ bin/dsconfig list-log-publishers --help
```

Use dsconfig in interactive mode

Running `dsconfig` in interactive command-line mode provides a menu-driven interface for accessing and configuring the PingData server. To start `dsconfig` in interactive mode, run the tool without any arguments:

```
$ bin/dsconfig
```

Running the tool requires server connection and authentication information. After connection information is confirmed, a menu of the available operation types is displayed.

Use dsconfig in non-interactive mode

Non-interactive command-line mode provides a simple way to make arbitrary changes to the server, and to use administrative scripts to automate configuration changes. To make changes to multiple configuration objects at the same time, use batch mode.

The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The `--no-prompt` argument specifies non-interactive mode. The `{globalArgs}` argument provides a set of arguments that specify how to connect and authenticate to the server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on the implementation. The `{subcommand}` specifies which general action to perform. The following uses standard LDAP connections:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

The following uses SASL GSSAPI (Kerberos) parameters:

```
$ bin/dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The `{subcommandArgs}` argument contains a set of arguments specific to the particular task. To always display the advanced properties, use the `--advanced` command-line option.

> **Note**
> Global arguments can appear anywhere on the command line. The subcommand-specific arguments can appear anywhere after the subcommand.

Use dsconfig batch mode

The `dsconfig` tool provides a batching mechanism that reads multiple invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The `--no-prompt` option is required with `dsconfig` in batch mode.

```
$ bin/dsconfig --no-prompt \
  --hostname host1 \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --batch-file /path/to/sync-pipe-config.txt
```

If a `dsconfig` command has a missing or incorrect argument, the command will fail and stop the batch process without applying any changes to the server. A `--batch-continue-on-error` option is available, which instructs `dsconfig` to apply all changes and skip any errors.

View the `logs/config-audit.log` file to review the configuration changes made to the server, and use them in the batch file. The batch file can have blank lines for spacing, and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The PingDataSync Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to Ping Identity PingDataSync Server machines.

Using the resync Tool

The resync tool provides bulk synchronization that can be used to verify the synchronization setup. The tool operates on a single Sync Pipe at a time, retrieves entries from the Sync Source in bulk, and compares the source entries with the corresponding destination entries. If destination entries are missing or attributes are changed, they are updated.

The command provides a --dry-run option that can be used to test the matches between the Sync Source and Destination, without committing any changes to the target topology. The resync tool also provides options to write debugging output to a log.

Note
The resync tool should be used for relatively small datasets. For large deployments, export entries from the Sync Source into an LDIF file, run the bin/translate-ldif tool to translate and filter the entries into the destination format, and then import the result LDIF file into the Sync Destination.

Use the resync --help command for more information and examples. Logging is located in logs/tools/resync.log and logs/tools/resync-errors.log. If necessary, the logging location can be changed with the --logFilePath option.

Testing Attribute and DN Maps

The resync tool can be used to test how attribute maps and DN maps are configured by synchronizing a single entry. If the --logFilePath and --logLevel options are specified, the resync tool generates a log file with details.

Use the--dry-run option and specify a single entry, such as uid=user.0. Any logging performed during the operation appears in logs/tools/resync.log.

```
$ bin/resync --pipe-name sun-to-ds-sync-pipe \
  --sourceSearchFilter "(uid=user.0)" \
  --dry-run \
  --logLevel debug
```

Verifying the Synchronization Configuration

The most common use for the resync tool is to test that the Sync Pipe configuration has been set up correctly. For example, the following procedure assumes that the configuration was set up with the Sync Source topology (two replicated Sun Directory servers) with 2003 entries; the Sync Destination topology (two replicated PingData PingDirectory Server) has only the base entry and the cn=Sync User entry. Both source and destination topologies have their LDAP Change Logs enabled. Because both topologies are not actively being updated, the resync tool can be run with one pass through the entries.

Use resync with the --dry-run option to check the synchronization configuration. The output displays a timestamp that can be tracked in the logs.

```
$ bin/resync --pipe-name sun-to-ds-sync-pipe \
  --numPasses 1 \
  --dry-run
```

```
Starting Pass 1

Status after completing all passes[20/Mar/2010:10:20:07 -0500]
----------------------------------
Source entries retrieved 2003
Entries missing 2002
Entries out-of-sync 1
Duration (seconds) 4

Resync completed in 4 s.

0 entries were in-sync, 0 entries were modified, 0 entries were created,
1 entries are still out-of-sync, 2002 entries are still missing, and
0 entries could not be processed due to an error
```

Populating an Empty Sync Destination Topology

The following procedure uses the `resync` tool to populate an empty Sync Destination topology for small datasets. For large deployments, use the `bin/translate-ldif`.

In this example, assume that the Sync Destination topology has only the base entry (`dc=example,dc=com`) and the `cn=Sync User` entry. Perform the following steps to populate an empty Sync Destination:

1. Run the `resync` command with the log file path and with the log level `debug`. Logging is located in `logs/tools/resync.log` and `logs/tools/resync-errors.log`.

   ```
   $ bin/resync --pipe-name sun-to-ds-sync-pipe \
     --numPasses 1 \
     --logLevel debug
   ```

2. Open the `logs/resync-failed-DNs.log` file in a text editor to locate the error and fix it. An entry cannot be created because the parent entry does not exist.

   ```
   # Entry '(see below)' was dropped because there was a failure at the
   resource:
   Failed to create entry uid=mlott,ou=People,dc=example,dc=com. Cause:
   LDAPException(resultCode=no such object, errorMessage='Entry
   uid=user.38,ou=People,dc=example,dc=com cannot be added because its parent
   entry ou=People,dc=example,dc=com does not exist in the server',
   matchedDN='dc=example,dc=com')
   (id=1893859385ResourceOperationFailedException.java:126 Build
   revision=4881)
   dn: uid=user.38,ou=People,dc=example,dc=com
   ```

3. Rerun the `resync` command. The command creates the entries in the Sync Destination topology that are present in the Sync Source topology.

   ```
   $ bin/resync --pipe-name sun-to-ds-sync-pipe
   ```

   ```
   ...(output from each pass)...

   Status after completing all passes[20/Mar/2016:14:23:33 -0500]
   ```

```
-----------------------------
Source entries retrieved 160
Entries in-sync 156
Entries created 4
Duration (seconds) 11

Resync completed in 12 s.

156 entries were in-sync, 0 entries were modified, 4 entries were created,
0 entries are still out-of-sync, 0 entries are still missing, and 0
entries could not be processed due to an error
```

Note
If importing a large amount of data into an PingData PingDirectory Server, run `export-ldif` and `import-ldif` on the newly populated backend for most efficient disk space use. If needed, run `dsreplication initialize` to propagate the efficient layout to additional replicas.

Setting the Synchronization Rate

The `resync` command has a `--ratePerSecondFile` option that enables a specific synchronization rate. The option can be used to adjust the rate during off-peak hours, or adjust the rate based on measured loads for very long operations.

Note
The `resync` command also has a `--ratePerSecond` option. If this option is not provided, the tool operates at the maximum rate.

Run the `resync` tool first at 100 operations per second, measure the impact on the source servers, then adjust as desired. The file must contain a single positive integer number surrounded by white space. If the file is updated with an invalid number, the rate is not updated.

1. Create a text file containing the rate. The number must be a positive integer surrounded by white space.

```
$ echo ’100 ’ > rate.txt
```

2. Run the `resync` command with the `--ratePerSecondFile` option.

```
$ bin/resync --pipe-name "sun-to-ds-sync-pipe" \
  --ratePerSecondPath rate.txt
```

Synchronizing a Specific List of DNs

The `resync` command enables synchronizing a specific set of DNs that are read from a file using the `--sourceInputFile` option. This option is useful for large datasets that require faster processing by targeting individual base-level searches for each source DN in the file. If any DN fails (parsing, search, or process errors), the command creates an output file of the skipped entries (`resync-failed-DNs.log`), which can be run again.

The file must contain only a list of DNs in LDIF format with `dn:` or `dn::`. The file can include comment lines. All DNs can be wrapped and are assumed to be wrapped on any lines that begin with a space followed by text. Empty lines are ignored.

Small files can be created manually. For large files, use `ldapsearch` to create an LDIF file, as follows:

1. Run an `ldapsearch` command using the special OID "1.1" extension, which only returns the DNs in the DIT. For example, on the Sync Source directory server, run the following command:

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com \
  --baseDN dc=example,dc=com \
  --searchScope sub "(objectclass=*)" "1.1" > dn.ldif
```

2. Run the `resync` command with the file.

```
$ bin/resync --pipe-name "sun-to-ds-pipe" \
  --sourceInputFile dn.ldif
```

```
Starting pass 1
[20/Mar/2016:10:32:11 -0500]

-------------------------------------
Resync pass 1
Source entries retrieved 1999
Entries created 981
Current pass, entries processed 981
Duration (seconds) 10
Average ops/second 98
Status after completing all passes[20/Mar/2016:10:32:18 -0500]

----------------------------------
Source entries retrieved 2003
Entries created 2003
Duration (seconds) 16
Average ops/second 98
Resync completed in 16 s.
0 entries were in-sync, 0 entries were modified, 2003 entries were
created, 0 entries are still out-of-sync, 0 entries are still missing, and
0 entries could not be processed due to an error
```

3. View the `logs/tools/resync-failed-DNs.log` to determine skipped DNs. Correct the source DNs file, and rerun the `resync` command.

Using the realtime-sync Tool

The `bin/realtime-sync` tool controls starting and stopping synchronization globally or for individual Sync Pipes. The tool also provides features to set a specific starting point for real-time synchronization.

To see the current status of real-time synchronization, view the monitor properties: num-sync-ops-in-flight, num-ops-in-queue, and source-unretrieved-changes. For example, use ldapsearch to view a specific Sync Pipe’s monitor information:

```
$ bin/ldapsearch --baseDN cn=monitor \
--searchScope sub "(cn=Sync PipeMonitor: PIPE_NAME)"
```

The Stats Logger can also be used to view status. See the *Ping IdentityPingDirectory Server Administration Guide* for details.

Starting Real Time Synchronization Globally

The realtime-sync tool assumes that the synchronization topology is configured correctly.

Perform the following steps to start real time synchronization globally:

1. Run the tool from the <server-root>/bin directory. This example assumes that a single Sync Pipe called "dsee-to-ds-sync-pipe" exists.

```
$ bin/realtime-sync start --pipe-name "dsee-to-ds-sync-pipe" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. If more than one Sync Pipe is configured, specify each using the --pipe-name option. The following example starts synchronization for a bidirectional synchronization topology.

```
$ bin/realtime-sync start --pipe-name "Sun DS to DS" \
  --pipe-name "DS to Sun DS" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Starting or Pausing Synchronization

Pause or start synchronization by using the start and stop subcommands. If synchronization is stopped and then restarted, changes made at the Sync Source while synchronization was stopped will still be detected and applied. Synchronization for individual Sync Pipes can be started or stopped using the --pipe-name argument. If the --pipe-name argument is omitted, then synchronization is started or stopped globally.

The following command stops all Sync Pipes:

```
$ bin/realtime-sync stop --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --no-prompt
```

If a topology has two Sync Pipes, Sync Pipe1 and Sync Pipe2, the following command stops Sync Pipe1.

```
$ bin/realtime-sync stop --pipe-name "Sync Pipe1" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

Setting Startpoints

Startpoints instruct the Sync Pipe to ignore all changes made prior to the current time. Once synchronization is started, only changes made after this command is run will be detected at the Sync Source and applied at the Sync Destination.

The `set-startpoint` subcommand is often run during the initial setup prior to starting realtime synchronization. It should be run prior to initializing the data in the Sync Destination.

The `set-startpoint` subcommand can start synchronization at a specific change log number, or back at a state that occurred at a specific time. For example, synchronization can start 10 minutes prior to the current time.

Perform the following steps to set a synchronization startpoint:

1. If started, stop the synchronization topology globally with the following command:

   ```
   $ bin/realtime-sync stop --pipe-name "Sync Pipe1" \
     --port 389 \
     --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret \
     --no-prompt
   ```

2. Set the startpoint for the synchronization topology. Any changes made before setting this command will be ignored.

   ```
   $ bin/realtime-sync set-startpoint --pipe-name "Sync Pipe1" \
     --port 389 \
     --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret \
     --no-prompt \
     --beginning-of-changelog
   ```

   ```
   Set StartPoint task 2011072109564107 scheduled to start immediately
   [21/Jul/2016:09:56:41 -0500] severity="INFORMATION" msgCount=0
   msgID=1889535170
   message="The startpoint has been set for Sync Pipe 'Sync Pipe1'.
   Synchronization will resume from the last change number in the Sync
   Source"
   Set StartPoint task 2011072109564107 has been successfully completed
   ```

Restarting Synchronization at a Specific Change Log Event

Perform the following steps to restart synchronization at a specific event:

1. Search for a specific change log event from which to restart the synchronization state. On one of the endpoint servers, run `ldapsearch` to search the change log.

```
$ bin/ldapsearch -p 1389
--bindDN "uid=admin,dc=example,dc=com" \
--bindPassword secret \
--baseDN cn=changelog \
--dontWrap
```

```
"(objectclass=*)"
dn: cn=changelog
objectClass: top
objectClass: untypedObject
cn: changelog

dn: changeNumber=1,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.13,ou=People,dc=example,dc=com
changeType: modify
changes::
cmVwbGFjZTogcm9vbU51bWJlcgpyb29tTnVtYmVyOiAwMTM4Ci0KcmVwbGFjZTogbW9kaW
ZpZXJzTmFtZQptb2RpZmllcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290
IEROcyxjbj1jb25maWcKLQpyZXBsYWNlOiBkcy11cGRhdGUtdGltZQpkcy11cGRhdGUtdG
ltZTo6IEFBQUJKZ25OWlUwPQotCgA=
changenumber: 1
          ... (more output)
dn: changeNumber=2329,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.49,ou=People,dc=example,dc=com
changeType: modify
changes::
cmVwbGFjZTogcm9vbU51bWJlcgpyb29tTnVtYmVyOiAwNDMzCi0KcmVwbGFjZTogbW9kaW
ZpZXJzTmFtZQptb2RpZmllcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290
IEROcyxjbj1jb25maWcKLQpyZXBsYWNlOiBkcy11cGRhdGUtdGltZQpkcy11cGRhdGUtdG
ltZTo6IEFBQUJKZ25OMC84PQotCgA=
changenumber: 2329
```

2. Restart synchronization from change number 2329 using the `realtime-sync` tool. Any event before this change number will not be synchronized to the target endpoint.

```
$ bin/realtime-sync set-startpoint \
  --change-number 2329 \
  --pipe-name "Sync Pipe 1" \
  --bindPassword secret \
  --no-prompt
```

Changing the Synchronization State by a Specific Time Duration

The following command will start synchronizing data at the state that occurred 2 hours and 30 minutes prior to the current time on External Server 1 for Sync Pipe 1. Any changes made before this time will not be synchronized. Specify days (d), hours (h), minutes (m), seconds (s), or milliseconds (ms).

Use `realtime-sync` with the `--startpoint-rewind` option to set the synchronization state and begin synchronizing at the specified time.

```
$ bin/realtime-sync set-startpoint \
  --startpoint-rewind 2h30m \
  --pipe-name "Sync Pipe 1" \
  --bindPassword secret \
  --no-prompt
```

Scheduling a Realtime Sync as a Task

The `realtime-sync` tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDataSync Server's process. To schedule an operation, supply LDAP connection options that allow this tool to communicate with the server through its task interface. Tasks can be scheduled to run immediately or at a later time. Once scheduled, tasks can be managed using the `manage-tasks` tool.

Perform the following steps to schedule a synchronization task:

1. Use the `--start` option with the `realtime-sync` command to schedule a start for the synchronization topology. The following command will set the start time at July 21, 2016 at 12:01:00 AM. The scheduled task can be stopped with the `--stop` subcommand.

   ```
   $ bin/realtime-sync set-startpoint \
     --pipe-name "sun-to-ds-sync-pipe" \
     --port 389 \
     --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret \
     --start 20150721000100 \
     --no-prompt
   ```

   ```
   Set StartPoint task 2009072016103807 scheduled to start Jul 21, 2016
   12:01:00 AM CDT
   ```

2. Run the `manage-tasks` tool to manage or cancel the task.

   ```
   $ bin/manage-tasks --port 7389 \
     --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret
   ```

Configuring the PingDirectory Server Backend for Synchronizing Deletes

The PingDirectory Server’s change log backend's `changelog-deleted-entry-include-attribute` property specifies which attributes should be recorded in the change log entry during a DELETE operation. Normally, the PingDataSync Server cannot correlate a deleted entry to the entry on the destination. If a Sync Class is configured with a filter, such as "include-filter: objectClass=person," the objectClass attribute must be recorded in the change log entry. Special correlation attributes (other than DN), will also need to be recorded on the change log entry to be properly synchronized at the endpoint server.

On each PingDirectory Server backend, use the dsconfig command to set the property.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set changelog-deleted-entry-include-attribute:objectClass
```

If the destination endpoint is an Oracle/Sun DSEE (or Sun DS) server, the Sun DSEE server does not store the value of the user deleting the entry, specified in the modifiers name attribute. It only stores the value of the user who last modified the entry while it still existed.

To set up a Sun DSEE destination endpoint to record the user who deleted the entry, use the Ping Identity Server SDK to create a plug-in as follows:

1. Update the Sun DSEE schema to include a deleted-by-sync auxiliary objectclass. It will only be used as a marker objectclass, and not require or allow additional attributes to be present on an entry.
2. Update the Sun DSEE Retro Change Log Plug-in to include the deleted-by-sync auxiliary object class as a value for the deletedEntryAttrs attribute.
3. Write an LDAPSyncDestinationPlugin script that in the preDelete() method modifies the entry that is being deleted to include the deleted-by-sync objectclass.
4. Update the Sync Class filter that is excluding changes by the Sync User to also include (!(objectclass=deleted-by-sync)).

Configure DN maps

Similar to attribute maps, DN maps define mappings when destination DNs differ from source DNs. These differences must be resolved using DN maps in order for synchronization to successfully take place. For example, the Sync Source could have a DN in the following format:

```
uid=jdoe,ou=People,dc=example,dc=com
```

While the Sync Destination could have the standard X.500 DN format.

DN mappings allow the use of wild cards for DN transformations. A single wild card (*) matches a single RDN component and can be used any number of times. The double wild card (**) matches zero or more RDN components and can be used only once.

Note
If a literal '*' is required in a DN then it must be escaped as '\2A'.

The wild card values can be used in the to-dn-pattern attribute using {1} to replace their original index position in the pattern, or {attr} to match an attribute value. For example:

```
*,**,dc=com->{1},ou=012,o=example,c=us
```

For example, using the DN, uid=johndoe,ou=People,dc=example,dc=com, and mapping to the target DN, uid=johndoe,ou=012,o=example,c=us:

- "*" matches one RDN component, uid=johndoe
- "**" matches zero or more RDN components, ou=People,dc=example
- "dc=com" matches dc=com in the DN.

The DN is mapped to the {1},ou=012,o=example,c=us. "{1}" substitutes the first wildcard element "uid=johndoe", so that the DN is successfully mapped to:

```
uid=johndoe,ou=012,o=example,c=us
```

Regular expressions and attributes from the user entry can also be used in the to-dn-pattern attribute. For example, the following expression constructs a value for the uid attribute, which is the RDN, out of the initials (first letter of givenname and sn) and the employee ID (the eid attribute) of a user.

```
uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid},{2},o=example
```

Note
The PingDataSync Server automatically validates any DN mapping prior to applying the configuration.

Configuring a DN Map Using dsconfig

A DN map can be configured using dsconfig, either with the interactive DN Map menu, or from the command line.

Perform the following to configure a DN map:

1. Use dsconfig to create a DN map for the PingDataSync Server.

```
$ bin/dsconfig --no-prompt create-dn-map \
  --map-name nested-to-flattened \
  --set "from-dn-pattern:*,*,dc=example,dc=com" \
  --set "to-dn-pattern:uid={givenname:/^(.)(.*)/\$1/s}{sn:/^(.)(.*)/\$1/s}
(eid},{2},o=example" \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. After DN mappings are configured, add the new DN map to a new Sync Class or modify an existing Sync Class.

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name test-sync-pipe \
  --class-name test-sync-class \
  --set dn-map:test-dn-map \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configure synchronization with JSON attribute values

The PingDataSync Server supports synchronization of attributes that hold JSON objects. The following scenarios are supported:

- **Synchronizing a JSON attribute to another JSON attribute** - A subset of fields can be synchronized, optionally retaining fields that appear at the destination but not at the source.
- **Synchronizing a JSON attribute to a non-JSON attribute** - A single field of the JSON value can be extracted with a constructed attribute mapping.
- **Synchronizing a non-JSON attribute to a JSON attribute** - The source value can be escaped so that it ensures the JSON value is properly formatted.
- **Attribute correlation** - A JSON field can be used when correlating a destination entry with a source entry.

The following examples show configuration scenarios based on the LDAP `ubidEmailJSON` attribute, which has fields of `value`, `type`, `primary`, and `verified`:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "primary" : true,
                "verified" : true}
```

Synchronize ubidEmailJSON fully

If a source JSON attribute value should be synchronized fully to a destination JSON attribute value, no special configuration is required.

Synchronize a subset of fields from the source attribute

For example, the following configuration can be used to synchronize the `value` and `type` fields of `ubidEmailJSON` from the source to a destination. To synchronize this source value:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "primary" : true}
```

to this value at the destination:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home"}
```

A JSON Attribute configuration object must be created and associated with the Sync Class. This can be done by either explicitly including the fields to synchronize:

```
$ bin/dsconfig create-json-attribute --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set include-field:type \
  --set include-field:value
```

Or by excluding the fields that should not be synchronized:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set exclude-field:preferred \
  --set exclude-field:verified
```

If the destination is prepared to only handle a specific subset of fields, then list the fields to include. However, if only a small, known subset of fields from the source should be excluded, then `exclude-field` could be used. In this example, the destination data for the `ubidEmailJSON` attribute will always be a subset of the full data.

> **Note**
> A Sync Class can be configured to exclude certain attributes from synchronization. Creating a regular attribute mapping will override this setting, and the attribute will be synchronized. Creating a JSON attribute mapping does not override this setting, and the JSON attribute will not be synchronized. A JSON attribute is not a traditional attribute mapping. It only includes information on the destination attribute name. To work around this, the attribute either needs to be mapped from a source attribute, or have its value constructed.

The following scenario illustrates how the destination can include additional fields that are not present at the source.

Retain destination-only fields

To synchronize changes to the source fields while preserving the value of the the `verified` field of the `ubidEmailJSON` attribute at the destination, configure the JSON Attribute as follows:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set id-field:value \
  --set exclude-field:verified
```

The `verified` field is excluded and `value` is chosen to correlate destination values with source values. For example, given that the source and destination `value` fields match, if the source initially contained:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home"}
```

and the destination contained:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "verified" : true},
```

if the source changed to:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "other"}
```

then the destination would change to:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "other",
                "verified" : true}
```

However, if the source changed to:

```
ubidEmailJSON: {"value" : "john.smith@example.com",
                "type" : "home"}
```

then the destination would be updated to:

```
ubidEmailJSON: {"value" : "john.smith@example.com",
                "type" : "home"}
```

The `verified` field has been dropped because this logically represents a new JSON object rather than an update of an existing one.

Synchronize a field of a JSON attribute into a non-JSON attribute

If the source stores:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home"}
```

but the destination stores:

```
mail: jsmith@example.com
```

To synchronize changes between these systems, a constructed attribute mapping must be configured:

```
$ bin/dsconfig create-attribute-mapping \
  --map-name "Attribute Map" \
  --mapping-name mail \
  --type constructed \
  --set "value-pattern:{ubidEmailJSON.value}"
```

The `value-pattern` syntax allows attributes to be referenced by placing them in `{}`. JSON fields within the attribute can be referenced by using the syntax `{attribute.field}`. See this property in the Configuration Refernce guide, or `dsconfig` tool command help for more information.

After the “Attribute Map” is created, it can be referenced from the Sync Class:

```
$ bin/dsconfig set-sync-class-prop
  --pipe-name "A to B" \
  --class-name Users \
  --set "attribute-map:Attribute Map"
```

Note

While LDAP attribute names are not case sensitive, the JSON field names are. By default, errors related to attribute mapping are not logged. To enable error logging, configure the Debug Logger with the following:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true
```

```
$ bin/dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name com.unboundid.directory.sync.mapping \
  --set debug-level:warning
```

Synchronize a non-JSON attribute into a field of a JSON attribute

This is the reverse of the previous example. Suppose the source stores:

```
mail: jsmith@example.com
```

but the destination stores:

```
ubidEmailJSON: {"value" : "jsmith@example.com"}
```

A constructed attribute mapping can be used in this case as well:

```
$ bin/dsconfig create-attribute-mapping \
  --map-name "Attr Map" \
  --mapping-name ubidEmailJSON \
  --type constructed \
  --set 'value-pattern:{{"value" : "{mail:jsonEscape}"}}'
```

When constructing the value, the following are important:

- Double curly brackets (`{{ }}`) are necessary to represent a single curly bracket (`{ }`) in the output. These brackets are typically used to reference attribute values.
- Attribute values that appear within a JSON attribute should be escaped using the `:jsonEscape` modifier. This prevents values that include quotes like `'“John Smith” <jsmith@example.com>’` from producing invalid JSON.

In this example, a JSON Attribute object should be created since the destination value is likely to be augmented with additional information:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set id-field:value \
  --set include-field:value
```

Correlating attributes based on JSON fields

When the destination of a Sync Pipe is a Ping Directory Server or PingDirectoryProxy Server, source and destination entries can be correlated by referencing a field within a JSON attribute. In the following example, source entries will be matched with destination entries that have the same `value` field within the `ubidEmailJSON` value.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name "A to B" \
  --class-name Users \
  --set destination-correlation-attributes:ubidEmailJSON.value
```

This could also be used with the previous example, which does not store `ubidEmailJSON.value` at the source but maps into it before correlating at the destination.

Configure fractional replication

The PingDataSync Server supports fractional replication to any server type. For example, if a replica only performs user authentications, the PingDataSync Server can be configured to propagate only the `uid` and `userpassword` password policy attributes, reducing the database size at the replica and the network traffic needed to keep this servers synchronized.

The following example configures a fractional replication, where the `uid` and `userPassword` attributes of all entries in the source topology are synchronized to the destination topology. Because the `uid` and `userPassword` attributes are present, the `objectclass` attribute must also be synchronized. The example assumes that a PingDataSync Server and external servers are configured and a Sync Pipe and Sync Class are defined, but realtime synchronization or bulk resync have not been performed.

Perform the following steps to configure fractional replication from the `dsconfig` interactive menu:

1. On the main menu, type the number corresponding to Sync Classes.
2. On the Sync Class menu, type the number corresponding to viewing and editing an existing Sync Class. Assume that only one Sync Class has been defined.
3. Verify that the Sync Pipe and Sync Class exist.
4. On the Sync Class Properties menu, type the number specifying the source LDAP filter (`include-filter` property) that defines which source entries are to be included in the Sync Class.
5. On the Include-Filter Property menu, type the number corresponding to adding a filter value. For this example, type (`objectclass=person`). When prompted, enter another filter. Press **Enter** to continue. On the menu, enter 1 to use the value when specifying it.
6. On the Sync Class Properties menu, type the number corresponding to the `auto-mapped-source-attribute` property. Change the value from "`-all-`" to a specific attribute, so that only the specified attribute is automatically mapped from the source topology to the destination topology.
7. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to adding the source attributes that will be automatically mapped to the destination attributes of the same name. When prompted, enter each attribute, and then press **Enter**.

```
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: uid
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: userPassword
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: objectclass
```

```
Enter another value for the 'auto-mapped-source-attribute' property
[continue]:
```

8. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to removing one or more values. In this example, remove the "`-all-`" value, so that only the `objectclass`, `uid`, and `userPassword` attributes are only synchronized.
9. On the Auto-Mapped-Source-Attribute Property menu, press **Enter** to accept the values.
10. On the Sync Class Properties menu, type the number corresponding to excluding some attributes from the synchronization process. When using the `objectclass=person` filter, the `cn`, `givenName`, and `sn` attributes must be excluded. Enter the option to add one or more attributes, and then add each attribute to exclude on the `excluded-auto-mapped-source-attributes` Property menu. For this example, exclude the `cn`, and `sn` attributes, which are required attributes of the `Person` objectclass. Also exclude the `givenName` attribute, which is an optional attribute of the `inetOrgPerson` objectclass.

```
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]: givenName
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]: sn
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]:
```

11. On the Excluded-Auto-Mapped-Source-Attributes Property menu, press **Enter** to accept the changes.

Note
If using `entryUUID` as a correlation attribute, some attribute uniqueness errors may occur while using the `resync` tool. Either set the `excluded-auto-mapped-source-attributes` property value to `entryUUID` on the Sync Class configuration menu, or run `resync` with the `--excludeDestinationAttr` entryUUID argument.

12. On the Sync Class Properties menu, review the configuration and accept the changes.
13. On the server instances in the destination topology, turn off schema checking to avoid a schema error that occurs when the required attributes in the `Person` object class are not present. Make sure that the global configuration property for the `server-group` is set to `all-servers`. Use the following command to turn off schema checking on all of the servers in the group.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set check-schema:false \
  --applyChangeTo server-group \
  --port 3389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

14. Run `bin/resync` to load the filtered data from the source endpoint to the target endpoint.

```
$ bin/resync --pipe-name "test-sync-pipe" \
  --numPasses 3
```

15. Run `bin/realtime-sync` to start synchronization.

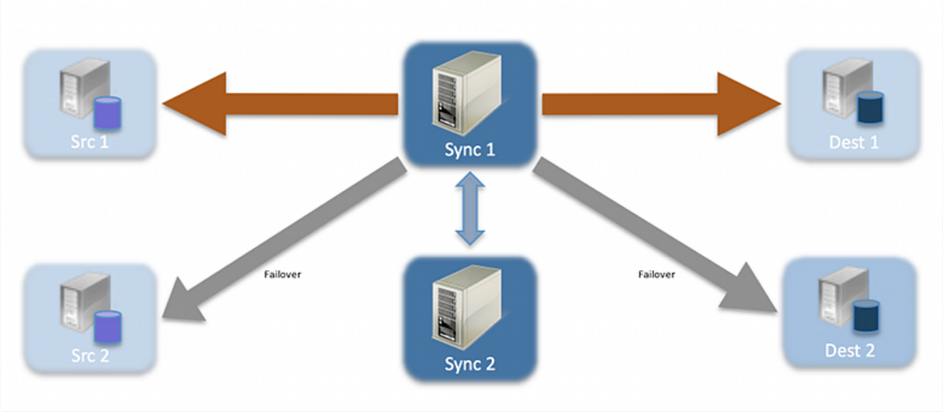
```
$ bin/realtime-sync start --pipe-name "test-sync-pipe" \
  --port 7389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --no-prompt
```

Configure failover behavior

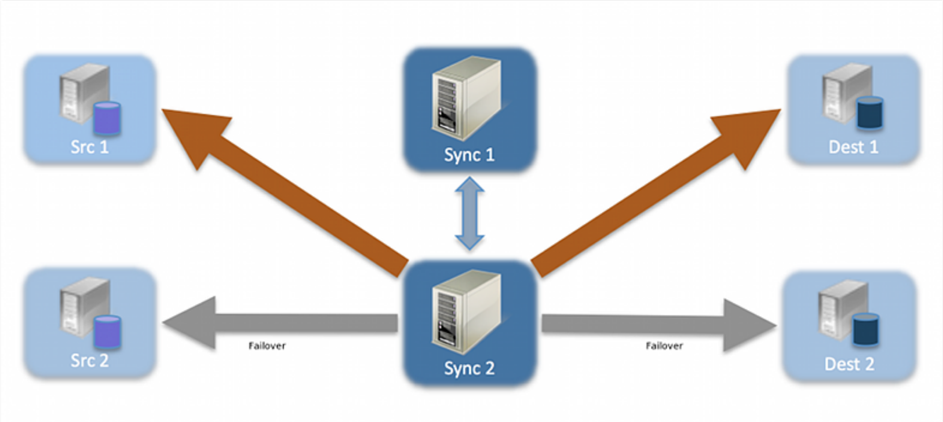
The following illustrates a simplified synchronization topology with a single failover server on the source, destination, and PingDataSync Server, respectively. The gray lines represent possible failover connections in the event the server is down. The external servers are prioritized so that src1 has higher priority than src2; dest1 has higher priority than dest2.

The main PingDataSync Server and its redundant failover instance communicate with each other over LDAP and bind using `cn=IntraSync User,cn=Root DNs,cn=config`. The servers run periodic health checks on each other and share information on all changes that have been processed. Whenever the failover server loses connection to the main server, it assumes that the main server is down and begins processing changes from the last known change. Control reverts back to the main server once it is back online.

Unlike the PingDataSync Server servers, the external servers and their corresponding failover server(s) do not run periodic health checks. If an external server goes offline, the failover server will receive transactions and remain connected to the PingDataSync Server until the Sync Pipe is restarted, regardless of if the main external server comes back online.



The PingDataSync Server in a Simplified Setup



The PingDataSync Server Sample Failover

Conditions that trigger immediate failover

Immediate failover occurs when the PingDataSync Server receives one of the following error codes from an external server:

- BUSY (51)
- UNAVAILABLE (52)
- SERVER CONNECTION CLOSED (81)
- CONNECT ERROR (91)

If the PingDataSync Server attempts a write operation to a target server that returns one of these error codes, the PingDataSync Server will automatically fail over to the next highest prioritized server instance in the target topology, issue an alert, and then reissue the retry attempt. If the operation is unsuccessful for any reason, the server logs an error.

Failover server preference

The PingDataSync Server supports endpoint failover, which is configurable using the `location` property on the external servers. By default, the PingDataSync Server prefers to connect to and failover to endpoint servers in the same location as itself. If no location settings are configured, the PingDataSync Server will iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

The PingDataSync Server does not do periodic health checks and will not failover to a preferred server automatically. Because of the cost of sync failover, it will always stay connected to a given server until that server stops responding or until the Sync Pipe is restarted. When a failover does happen, it will always go back to the most preferred server (optionally using location settings to determine this) and work its way down the list. The following provides an example configuration of external servers:

```
austin1.server.com:1389
london1.server.com:2389
boston1.server.com:3389
```

```
austin2.server.com:4389
boston2.server.com:5389
london2.server.com:6389
```

If the austin1 server were to become unavailable, the PingDataSync Server will automatically pick up changes on the next server on the list, london1. If london1 is also down, then the next server, boston1 will be picked up. Once the PingDataSync Server iterates through the list, it returns to the top of the list. So, if the PingDataSync Server is connected to london2 and it goes down, it will fail over to austin1.

To minimize WAN traffic, configure the `location` property for each external server using the `dsconfig` command on the PingDataSync Server. Assume that PingDataSync Server has its own `location` property (set in the Global Configuration) set to "austin."

```
austin1.server.com:1389 location=austin
london1.server.com:2389 location=london
boston1.server.com:3389 location=boston
austin2.server.com:4389 location=austin
boston2.server.com:5389 location=boston
london2.server.com:6389 location=london
```

With the `location` property set for each server, the PingDataSync Server gets its changes from server austin1. If austin1 goes down, the PingDataSync Server will pick up changes from austin2. If austin2 goes down, the server will iterate through the rest of the list in the order it is configured.

The `location` property has another sub-property, `preferred-failover-location` that specifies a set of alternate locations if no servers in this location are available. If multiple values are provided, servers are tried in the order in which the locations are listed. The `preferred-failover-location` property provides more control over the failover process and allows the failover process to jump to a specified location. Care must be used so that circular failover reference does not take place. Here is an example configuration:

```
austin1.server.com:1389 location=austin preferred-failover-location=boston
london1.server.com:2389 location=london preferred-failover-location=austin
boston1.server.com:3389 location=boston preferred-failover-location=london
austin2.server.com:4389 location=austin preferred-failover-location=boston
boston2.server.com:5389 location=boston preferred-failover-location=austin
london2.server.com:6389 location=london preferred-failover-location=london
```

The PingDataSync Server will respect the `preferred-failover-location` if it cannot find any external servers in the same location as itself, it will look for any external servers in its own `preferred-failover-location`. In this example, when austin1 becomes unavailable, it will fail over to austin2 because they are in the same location. If austin2 is unavailable, it will fail over to boston1, which is in the `preferred-failover-location` of the PingDataSync Server. If boston1 in unavailable, the PingDataSync Server will fail over to boston2, and finally, it will try the london1 and london2 servers.

Configuration properties that control failover behavior

There are four important advanced properties to fine tune the failover mechanism:

- `max-operation-attempts` (Sync Pipe)
- `response-timeout` (source and destination endpoints)
- `max-failover-error-code-frequency` (source and destination endpoints)
- `max-backtrack-replication-latency` (source endpoints only)

These properties apply to the following LDAP error codes:

LDAP Error Codes

Error Code	Description
ADMIN_LIMIT_EXCEEDED (11)	Indicates that processing on the requested operation could not continue, because an administrative limit was exceeded.
ALIAS_DEREFERENCING_ PROBLEM (36)	Indicates that a problem was encountered while attempting to dereference an alias for a search operation.
CANCELED (118)	Indicates that a cancel request was successful, or that the specified operation was canceled.
CLIENT_SIDE_LOCAL_ERROR (82)	Indicates that a local (client-side) error occurred.
CLIENT_SIDE_ENCODING_ERROR (83)	Indicates that an error occurred while encoding a request.
CLIENT_SIDE_DECODING_ERROR (84)	Indicates that an error occurred while decoding a request.
CLIENT_SIDE_TIMEOUT (85)	Indicates that a client-side timeout occurred.
CLIENT_SIDE_USER_CANCELLED (88)	Indicates that a user canceled a client-side operation.
CLIENT_SIDE_NO_MEMORY (90)	Indicates that the client could not obtain enough memory to perform the requested operation.
CLIENT_SIDE_CLIENT_LOOP (96)	Indicates that a referral loop is detected.
CLIENT_SIDE_REFERRAL_LIMIT_ EXCEEDED (97)	Indicates that the referral hop limit was exceeded.
DECODING_ERROR (84)	Indicates that an error occurred while decoding a response.
ENCODING_ERROR (83)	Indicates that an error occurred while encoding a response.
INTERACTIVE_TRANSACTION_ ABORTED (30221001)	Indicates that an interactive transaction was aborted.
LOCAL_ERROR (82)	Indicates that a local error occurred.
LOOP_DETECT (54)	Indicates that a referral or chaining loop was detected while processing a request.
NO_MEMORY (90)	Indicates that not enough memory could be obtained to perform the requested operation.
OPERATIONS_ERROR (1)	Indicates that an internal error prevented the operation from being processed properly.
OTHER (80)	Indicates that an error occurred that does not fall into any of the other categories.
PROTOCOL_ERROR (2)	Indicates that the client sent a malformed or illegal

LDAP Error Codes

Error Code	Description
	request to the server.
TIME_LIMIT_EXCEEDED (3)	Indicates that a time limit was exceeded while attempting to process the request.
TIMEOUT (85)	Indicates that a timeout occurred.
UNWILLING_TO_PERFORM (53)	Indicates that the server is unwilling to perform the requested operation.

The max-operation-attempts property

The `max-operation-attempts` property (part of the Sync Pipe configuration) specifies the maximum number of times to retry a synchronization operation that fails for reasons other than the Sync Destination being busy, unavailable, server connection closed, or connect error.

To change the default number of retries, use `dsconfig` in non-interactive mode to change the `max-operation-attempts` value on the Sync Pipe object. The following command changes the number of maximum attempts from five (default) to four.

```
$ bin/dsconfig set-sync-pipe-prop \
  --pipe-name "Test Sync Pipe" \
  --set max-operation-attempts:4
```

The response-timeout property

The `response-timeout` property specifies how long the PingDataSync Server should wait for a response from a search request to a source server before failing with LDAP result code 85 (client-side timeout). When a client-side timeout occurs, the Sync Source will retry the request according to the `max-failover-error-code-frequency` property before failing over to a different source server and performing the retry. The total number of retries will not exceed the `max-operation-attempts` property defined in the Sync Pipe configuration. A value of zero indicates that there should be no client-side timeout. The default value is one minute.

Assuming a bidirectional topology, the property can be set with `dsconfig` on the Sync Source and Sync Destination, respectively.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name src \
  --set "response-timeout:8 s"
```

```
$ bin/dsconfig set-sync-destination-prop \
  --destination-name U4389 \
  --set "responsetimeout:9 s"
```

The max-failover-error-code-frequency property

The `max-failover-error-code-frequency` property (part of the Sync Source configuration) specifies the maximum time period that an error code can re-appear until it fails over to another server instance. This property allows the retry logic to be tuned, so that retries can be performed once on the same server before giving up and trying another server. The value can be set to zero if there is no acceptable error code frequency and failover should happen immediately. It can also be set to a very small value (such as 10 ms) if a high frequency of error codes is tolerable. The default value is three minutes.

To change the `max-failover-error-code-frequency` property, use `dsconfig` in non-interactive mode to change the property on the Sync Source object. The following command changes the frequency from three minutes to two minutes.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name source1 \
  --set "max-failover-error-code-frequency:2 m"
```

The max-backtrack-replication-latency property

The `max-backtrack-replication-latency` property (part of the Sync Source configuration) sets the time period that an PingDataSync Server will look for missed changes in the change log due to replication delays. The property should be set to a conservative upper-bound of the maximum replication delay between two servers in the topology. A value of zero implies that there is no limit on the replication latency. The default value is two hours. ThePingDataSync Server stops looking in the change log once it finds a change that is older than the maximum replication latency than the last change it processed on the failed server.

For example, after failing over to another server, the PingDataSync Server must look through the new server’s change log to find the equivalent place to begin synchronizing changes. Normally, the PingDataSync Server can successfully backtrack with only a few queries of the directory, but in some situations, it might have to look further back through the change log to make sure that no changes were missed. Because the changes can come from a variety of sources (replication, synchronization, and over LDAP), the replicated changes between directory servers are interleaved in each change log. When the PingDataSync Server fails over between servers, it has to backtrack to figure out where synchronization can safely pick up the latest changes.

Backtracking occurs until the following:

- The server determines that there is no previous change log state available for any source servers, so it must start at the beginning of the change log.
- The server finds the last processed replication change sequence number (CSN) from the last time it was connected to that replica, if at all. This process is similar to the `set-startpoint` functionality on the `realtime-sync` tool.
- The server finds the last processed replication CSN from every replica that has produced a change so far, and it determines that each change entry in the next oldest batch of changes has already been processed.
- The server finds a change that is separated by more than a certain duration (specified by the `max-backtrack-replication-latency` property) from the most recently processed change.

The following command changes the maximum backtracking from two hours to three hours.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name source1 \
  --set "max-backtrack-replication-latency:3 h"
```

Configure traffic through a load balancer

If a PingData server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request. If the actual client's IP address should be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and the PingData server. See the product documentation for the device type. For PingData servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP) and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

Configure authentication with a SASL external certificate

By default, the PingDataSync Server authenticates to the PingDirectory Server using LDAP simple authentication (with a bind DN and a password). However, the PingDataSync Server can be configured to use SASL EXTERNAL to authenticate to the PingDirectory Server with a client certificate.

> **Note**
> This procedure assumes that PingDataSync Server instances are installed and configured to communicate with the backend PingDirectory Server instances using either SSL or StartTLS.

After the servers are configured, perform the following steps to configure SASL EXTERNAL authentication:

1. Create a JKS keystore that includes a public and private key pair for a certificate that the PingDataSync Server instance(s) will use to authenticate to the PingDirectory Server instance(s). Run the following command in the instance root of one of the PingDataSync Server instances. When prompted for a keystore password, enter a strong password to protect the certificate. When prompted for the key password, press **ENTER** to use the keystore password to protect the private key:

   ```
   $ keytool -genkeypair \
     -keystore config/sync-user-keystore \
   ```

```
  -storetype JKS \
  -keyalg RSA \
  -keysize 2048 \
  -alias sync-user-cert \
  -dname "cn=Sync User,cn=Root DNs,cn=config" \
  -validity 7300
```

2. Create a `config/sync-user-keystore.pin` file that contains a single line that is the keystore password provided in the previous step.
3. If there are other PingDataSync Server instances in the topology, copy the `sync-user-keystore` and `sync-user-keystore.pin` files into the config directory for all instances.
4. Use the following command to export the public component of the user certificate to a text file:

```
$ keytool -export \
  -keystore config/sync-user-keystore \
  -alias sync-user-cert \
  -file config/sync-user-cert.txt
```

5. Copy the `sync-user-cert.txt` file into the `config` directory of all PingDirectory Server instances. Import that certificate into each server's primary trust store by running the following command from the server root. When prompted for the keystore password, enter the password contained in the `config/truststore.pin` file. When prompted to trust the certificate, enter **yes**.

```
$ keytool -import \
  -keystore config/truststore \
  -alias sync-user-cert \
  -file config/sync-user-cert.txt
```

6. Update the configuration for each PingDataSync Server instance to create a new key manager provider that will obtain its certificate from the `config/sync-user-keystore` file. Run the following `dsconfig` command from the server root:

```
$ dsconfig create-key-manager-provider \
  --provider-name "Sync User Certificate" \
  --type file-based \
  --set enabled:true \
  --set key-store-file:config/sync-user-keystore \
  --set key-store-type:JKS \
  --set key-store-pin-file:config/sync-user-keystore.pin
```

7. Update the configuration for each LDAP external server in each PingDataSync Server instance to use the newly-created key manager provider, and also to use SASL EXTERNAL authentication instead of LDAP simple authentication. Run the following `dsconfig` command:

```
$ dsconfig set-external-server-prop \
  --server-name ds1.example.com:636 \
  --set authentication-method:external \
  --set "key-manager-provider:Sync User Certificate"
```

After these changes, the PingDataSync Server should re-establish connections to the LDAP external server and authenticate with SASL EXTERNAL. Verify that the PingDataSync Server is still able to communicate with all backend servers by running the `bin/status` command. All of the servers listed in the "--- LDAP External Servers ---" section should have a status of `Available`. Review the PingDirectory Server access log can to make sure that the BIND RESULT log messages used to authenticate the connections from the PingDataSync Server include `authType="SASL", saslMechanism="EXTERNAL", resultCode=0`, and `authDN="cn=Sync User,cn=Root DNs,cn=config"`.

Server SDK extensions

Custom server extensions can be created with the Server SDK. Extension bundles are installed from a .zip archive or a file system directory. Use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note
The `manage-extension` tool must be used with Java extensions packaged using the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation.

The Server SDK enables creating extensions for all PingData servers. Cross-product extensions include:

- Access Loggers
- Alert Handlers
- Error Loggers
- Key Manager Providers
- Monitor Providers
- Trust Manager Providers
- OAuth Token Handlers
- Manage Extension Plugins

Chapter 4: Synchronizing with PingOne

The PingDataSync Server supports full synchronization for newly created or modified accounts with native password changes between directory server, relational databases, and PingOne systems.

This chapter presents configuration procedures for synchronization between PingDirectory Server, Nokia 8661 Directory Server, or other LDAP source servers or targets with PingOne.

Topics include:

Configuration on PingOne

You must have a PingOne account. The following need to be configured in the PingOne administrative console:

1. Log into PingOne.
2. On the **Connections** tab, create an application with the following settings:
    a. Make the application non-interactive.

    b. Provide a name, such as SyncApp.

    c. Grant the following scopes:

    - "p1:read:env:user"
    - "p1:create:env:user"
    - "p1:update:env:user"
    - "p1:delete:env:user"
    - "p1:read:env:population"
    - "p1:create:env:population"
    - "p1:update:env:population"
    - "p1:delete:env:population"
    - "p1:set:env:userPassword"
    - "p1:validate:env:userPassword"
3. Enable the application.

After the application is enabled, collect the following information to configure the PingDataSync Server:

- Client ID.
- Client secret.
- Token endpoint.
- PingOne environment ID.
- Organization ID.
- Default population ID.

Overview of configuration tasks

The PingDataSync Server provides a `dsconfig` batch file that can be configured and run to enable synchronization with PingOne. Included in the file, the following steps are used to configure synchronization with PingOne systems:

- **Configure the external server** – PingOne must be configured as an external server.
- **Define the Sync Source and Destination** – The PingOne environment name, OAuth client ID, token URL, and OAuth client secret are required to configure synchronization with a PingDataSync Server.
- **Create a Sync Pipe** – The `create-sync-pipe-config` tool is used to configure the Sync Pipes to communicate with the PingOne source or target.
- **Define attribute mappings** – PingOne has a fixed schema. In order to successfully synchronize users from a Sync Source, it is necessary to provide values for the following attributes:
  - `name`: A JSON object in the form `{"name": {"given": "John", "family": "Doe", "middle": "M"}}`.
  - `email`: A JSON string that must be a valid email address AND be UNIQUE in a PingOne environment. This value is required on creates.
  - `username`: A JSON string that must be UNIQUE in a PingOne environment. This value is required on creates.
  - `population`: A JSON object in the form `{"id": "population-id"}`. This value is required on creates.
  - `phone`: A JSON string in the form `"+1.5555555555"`.
- **Create a Sync Class** – After defining the attribute map, create a Sync Class and associate the map.
- **Define JSON attributes** – The name and population attributes must have the associated Sync Pipe and Sync Class.

The `reference-pingone-sync-destination-configuration.dsconfig` file can be found in the `<server-root>/resource` directory. Add the relevant configuration details to this file before running it.

Note

If values are provided for fields not listed here it may result in errors during synchronization. The `auto-mapped-source-attribute` must be set to `none`, because only the listed attributes can be synchronized. Passwords can be synchronized only when created. They cannot be synchronized when modified.

Configure synchronization

Use the `reference-pingone-sync-destination-configuration.dsconfig` file to configure PingOne synchronization. This file shows an example configuration for a one-way sync relationship between PingDirectory Server and PingOne. Configuration includes the following tasks.

Create the external server for source

The following sample configures a PingDirectory Server.

```
$ bin/dsconfig create-external-server \
  --server-name [SOURCE_SERVER_NAME] \
  --type ping-identity-ds \
  --set server-host-name:[SOURCE_HOST_NAME] \
  --set server-port:[SOURCE_PORT] \
  --set authentication-method:simple \
  --set bind-dn:[SOURCE_BIND_DN] \
  --set password:[SOURCE_BIND_PASSWORD]
```

Define Sync Source and Destination

The following commands create the Sync Source and destinations.

```
$ bin/dsconfig create-sync-source \
  --source-name DS  \
  --type ping-identity  \
  --set base-dn:dc=example,dc=com  \
  --set server:[SOURCE_SERVER_NAME]
```

```
$ bin/dsconfig create-sync-destination \
  --destination-name PingOne  \
  --type ping-one-customer  \
  --set ping-one-api-url:https://api.pingone.com/v1 \
  --set ping-one-auth-url:https://auth.pingone.com/v1/[PING_ONE_ENV_ID]
/as/token \
  --set ping-one-environment-id:[PING_ONE_ENV_ID]  \
  --set ping-one-oauth-client-id:[PING_ONE_OAUTH_CLIENT_ID]  \
  --set ping-one-oauth-client-secret:[PING_ONE_OAUTH_CLIENT_SECRET]
```

Create a Sync Pipe

The following sample command creates a Sync Pipe.

```
$ bin/dsconfig create-sync-pipe \
  --pipe-name DS_to_PingOne  \
  --set 'retry-backoff-increase-by:100 ms'  \
  --set sync-destination:PingOne  \
  --set sync-source:DS
```

Define attribute mappings

PingOne has a fixed schema. The `name` and `population` attributes are JSON objects in PingOne, therefore it is important to define them as JSON attributes in the Sync Class so that modifications are handled properly.

To successfully synchronize users from a Sync Source, values for the following attributes must be provided:

- `"name"`: A JSON object in the form `{"name": {"given": "John", "family": "Doe", "middle": "M"}}`
- `"email"`: A JSON string that must be a valid email address AND be UNIQUE in a PingOne Environment. This value is required on creates.
- `"username"`: A JSON string that must be UNIQUE in a PingOne Environment. This value is required on creates.
- `"population"`: A JSON object in the form `{"id": "population-id"}`. This value is required on creates.
- `"phone"`: A JSON string in the form `"+1.5555555555"`.

If values are provided for fields not listed here, errors may occur during synchronization. The following samples create the map:

```
$ bin/dsconfig create-attribute-map \
  --map-name PingOneUserMap
```

```
$ bin/dsconfig create-attribute-mapping \
  --map-name PingOneUserMap \
  --mapping-name username \
  --type constructed \
  --set value-pattern:{givenname}.{sn}
```

```
$ bin/dsconfig create-attribute-mapping \
  --map-name PingOneUserMap \
  --mapping-name password \
  --type direct \
  --set from-attribute:userPassword
```

```
$ bin/dsconfig create-attribute-mapping \
  --map-name PingOneUserMap \
  --mapping-name email \
  --type constructed \
  --set value-pattern:{givenname}.{sn}@example.com
```

```
$ bin/dsconfig create-attribute-mapping \
  --map-name PingOneUserMap \
  --mapping-name population \
  --type constructed  \
  --set 'value-pattern:{{"id":"[DEFAULT_POPULATION_ID]"}}'
```

```
$ bin/dsconfig create-attribute-mapping \
  --map-name PingOneUserMap \
  --mapping-name phone \
  --type direct \
  --set from-attribute:telephoneNumber
```

```
$ bin/dsconfig create-attribute-mapping \
  --map-name PingOneUserMap \
  --mapping-name name \
  --type constructed \
  --set 'value-pattern:{{"givenName":"{givenname:jsonEscape}","familyName":"
{sn:jsonEscape}"}}'
```

Create a Sync Class and associate the defined Attribute Map

```
$ bin/dsconfig create-sync-class \
  --pipe-name [SYNC_PIPE_NAME] \
  --class-name [SYNC_CLASS_NAME] \
  --set attribute-map:PingOneUserMap \
  --set destination-correlation-attributes:username \
  --set destination-correlation-attributes:email \
  --set creates-as-modifies:true \
  --set include-filter:(objectClass=person)
```

Define JSON attributes

```
$ bin/dsconfig create-json-attribute \
  --pipe-name [SYNC_PIPE_NAME] \
  --class-name [SYNC_CLASS_NAME] \
  --attribute-name name
```

```
$ bin/dsconfig create-json-attribute \
  --pipe-name [SYNC_PIPE_NAME] \
  --class-name [SYNC_CLASS_NAME] \
  --attribute-name population \
  --set include-field:id
```

Run the resync command

If `resync` is run against an instance that has already been loaded with users, the `password` attribute should be added to the list of ignored destination attributes.

$ resync --pipe-name [SYNC_PIPE_NAME] \
  --excludeDestinationAttr password

Setup debug targets for added logging

The following will log all of the raw HTTP traffic between PingOne and PingDataSync Server, which can include sensitive information, such as passwords and authentication tokens. Do not enable or run this on in a production environment.

```
$ bin/dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
com.unboundid.directory.sync.pingone.PingOneCustomerSyncDestination \
  --set debug-level:verbose
```

```
$ bin/dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name com.unboundid.directory.sync.pingone.HTTPTrafficLogger \
  --set debug-level:verbose
```

Chapter 5: Synchronizing with Active Directory systems

The PingDataSync Server supports full synchronization for newly created or modified accounts with native password changes between directory server, relational databases, and Microsoft Active Directory systems.

This chapter presents configuration procedures for synchronization between PingDirectory Server, Nokia 8661 Directory Server, or other LDAP source servers or targets with Microsoft Active Directory systems.

Topics include:

Overview of configuration tasks

To configure synchronization with Active Directory systems, the following tasks are performed:

- **Enable SSL connections** – If synchronizing passwords between systems, synchronization with Microsoft Active Directory systems requires that SSL be enabled on the Active Directory domain controller, so that the PingDataSync Server can securely propagate the `cn=Sync User` account password and other user passwords to the target.
- **Run the create-sync-pipe-config tool** – On the PingDataSync Server, use the `create-sync-pipe-config` tool to configure the Sync Pipes to communicate with the Active Directory source or target.
- **Configure outbound password synchronization on an PingDirectory Server Sync Source** – After running the `create-sync-pipe-config` tool, determine if outbound password synchronization from an PingDirectory Server Sync Source is required. If so, enable the Password Encryption component on all PingDirectory Server sources that receive password modifications. The PingDirectory Server uses the Password Encryption component, analogous to the Password Sync Agent component, to intercept password modifications and add an encrypted attribute, `ds-changelog-encrypted-password`, to the changelog entry. The component enables passwords to be synchronized securely to the Active Directory system, which uses a different password storage scheme. The encrypted attribute appears in the change log and is synchronized to the other servers, but does not appear in the entries.
- **Configure outbound password synchronization on an Active Directory Sync Source** – After running the `create-sync-pipe-config` tool, determine if outbound password synchronization from an Active Directory Sync Source is required. If so, install the Password Sync Agent (PSA) after configuring the PingDataSync Server.
- **Run the realtime-sync set-startpoint tool** – The `realtime-sync set-startpoint` command may take several minutes to run, because it must issue repeated searches of the Active Directory domain controller until it has paged through all the changes and receives a cookie that is up-to-date.

Configuring synchronization with Active Directory

The following procedure configures a one-way Sync Pipe with the Active Directory topology as the Sync Source and an PingDirectory Server topology as the Sync Destination.

1. From the server-root directory, start the PingDataSync Server.

```
$ <server-root>/bin/start-server
```

2. Run the `create-sync-pipe-config` tool to set up the initial synchronization topology.

   ```
   $ bin/create-sync-pipe-config
   ```

3. On the Initial Synchronization Configuration Tool menu, press **Enter** to continue the configuration.
4. On the Synchronization Mode menu, press **Enter** to select Standard mode.
5. On the Synchronization Directory menu, select the option for one-way (1) or bidirectional (2) for the synchronization topology.
6. On the Source Endpoint Type menu, enter the option for Microsoft Active Directory.
7. On the Source Endpoint Name menu, type a name for the source server, or accept the default.
8. On the Server Security menu, select the security connection type for the source server.
9. On the Servers menu, enter the host name and listener port for the Source Server, or press **Enter** to accept the default (port 636). The server will attempt a connection to the server. After adding the first server, add additional servers for the source endpoints, which will be prioritized below the first server.
10. On the Synchronization User Account DN menu, enter the User Account DN for the source servers. The account will be used exclusively by the PingDataSync Server to communicate with the source external servers. Enter a User Account DN and password. The User Account DN password must meet the minimum password requirements for Active Directory domains.
11. Set up the Destination Endpoint servers.

The Active Directory Sync User account

The Sync User created for Active Directory is added to the `cn=Administrators` branch and is given most of a root user's permissions. If this account cannot be secured and there is a need to configure the permissions required by the Sync User, the following are required to perform synchronization tasks:

As a Sync Source, these permissions are needed:

- List contents
- Read all properties
- Read permissions

Deleted items are a special case. For the Sync Server to see deleted entries, the user account must have sufficient access to `cn=Deleted Objects,<domain name>`. Giving access to that DN requires using the `dsacls` tool, such as:

```
# Take ownership may be required to make the needed changes.
dsacls "CN=Deleted Objects,DC=example,DC=com" /takeOwnership
```

```
# Give the Sync User generic read permission to the domain.
dsacls "CN=Deleted Objects,DC=example,DC=com" /G "example\SyncUser":GR
```

```
# List the permisison for the domain.
dsacls "CN=Deleted Objects,DC=example,DC=com"
```

To revoke all permissions from the Sync User, run the following `dsacls` command:

```
dsacls "CN=Deleted Objects,DC=example,DC=com" /R "example\SyncUser"
```

If Active Directory is used as a destination for synchronization, the Sync User account should not be changed.

Prepare external servers

Perform the following steps to prepare external servers:

1. After configuring the source and destination endpoints, the PingDataSync Server prompts to "prepare" each external server. The process requires trusting the certificate presented to the server, and then testing the connection. If this step is not performed, the process can be completed after configuring the Sync Pipes using the `prepare-endpoint-server` tool.
2. Configuring this server for synchronization requires manager access. Enter the DN and password of an account capable of managing the external directory server.
3. Enter the maximum age of changelog entries. The value is formatted as `[number][time-unit]`, where the time unit format resembles ("8h" for eight hours, "3d" for three days, "1w" for one week). Setting this value caps how long the PingDataSync Server can be offline. A smaller value limits how many changes are stored and is necessary to limit excessive changelog growth in high-modification environments.
4. To prepare another server in the topology, follow the prompts. The previously entered manager credentials can be reused to access additional servers. Repeat the process for each server configured in the system.

Configure Sync Pipes and Sync Classes

Perform the following steps to configure Sync Pipes and Sync Classes:

1. On the Sync Pipe Name menu, type a unique name to identify the Sync Pipe, or accept the default.
2. On the Pre-Configured Sync Class Configuration for Active Directory Sync Source menu, enter **yes** to synchronize user CREATE operations, and enter the object class for the user entries at the destination server, or accept the default (`user`). To synchronize user MODIFY and DELETE operations from Active Directory, enter **yes**.

3. To synchronize passwords from Active Directory, press **Enter** to accept the default (yes). If synchronizing passwords from Active Directory, install the Ping Identity Password Sync Agent component on each domain controller.
4. To create a DN map for the user entries in the Sync Pipe, enter the base DN for the user entries at the Microsoft Active Directory Sync Source, then enter the base DN for the user entries at the PingDataSync Server Destination.
5. A list of basic attribute mappings from the Microsoft Active Directory Source to the PingDirectory Server destination is displayed. More complex attribute mappings involving constructed or DN attribute mappings must be configured with the `dsconfig` tool. The following is a sample mapping.

```
Below is a list of the basic mappings that have been set up for user
entries synchronized from Microsoft Active Directory -> PingDirectory
Server. You can add to or modify this list with any direct attribute
mappings. To set up more complex mappings (such as constructed or DN
attribute mappings), use the 'dsconfig' tool.

1) cn -> cn
2) sn -> sn
3) givenName -> givenName
4) description -> description
5) sAMAccountName -> uid
6) unicodePwd -> userPassword
```

6. Enter the option to add a new attribute mapping. Enter the source attribute, and then enter the destination attribute. The following example maps the `telephoneNumber` attribute (Active Directory) to the `otherTelephone` attribute (PingDirectory Server).

```
Select an attribute mapping to remove, or choose 'n' to add a new one
[Press ENTER to continue]: n

Enter the name of the source attribute: telephoneNumber
Enter the name of the destination attribute: otherTelephone
```

7. If synchronizing group CREATE, MODIFY, and DELETE operations from Active Directory, enter **yes**.
8. Review the basic user group mappings.
9. On the Sync Pipe Sync Class Definitions menu, enter another name for a new Sync Class if required. Repeat steps 2–7 to define this new Sync Class. If no additional Sync Class definitions are required, press **Enter** to continue.
10. Review the Sync Pipe Configuration Summary, and accept the default ("write configuration"), which records the commands in a batch file (`sync-pipe-cfg.txt`). The batch file can be used to set up other topologies. The following summary shows two Sync Pipes and its associated Sync Classes.

```
>>>> Configuration Summary

  Sync Pipe: AD to PingDirectory Server
    Source: Microsoft Active Directory
      Type: Microsoft Active Directory
      Access Account: cn=Sync
User,cn=Users,DC=adsync,DC=PingIdentity,DC=com
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Servers: 10.5.1.149:636

    Destination: PingDirectory Server
      Type: PingDirectory Server
      Access Account: cn=Sync User,cn=Root DNs,cn=config
      Base DN: dc=example,dc=com
      Servers: localhost:389

    Sync Classes:
      Microsoft Active Directory Users Sync Class
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Filters: (objectClass=user)
      DN Map: **,CN=Users,DC=adsync,DC=PingIdentity,DC=com ->{1},ou=users,
      dc=example,dc=com
      Synchronized Attributes: Custom set of mappings are defined
      Operations: Creates,Deletes,Modifies

  Sync Pipe: PingDirectory Server to AD
    Source: PingDirectory Server
      Type: PingDirectory Server
      Access Account: cn=Sync User,cn=Root DNs,cn=config
      Base DN: dc=example,dc=com
      Servers: localhost:389

    Destination: Microsoft Active Directory
      Type: Microsoft Active Directory
      Access Account: cn=Sync
User,cn=Users,DC=adsync,DC=PingIdentity,DC=com
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Servers: 10.5.1.149:636

    Sync Classes:
       PingDirectory Server Users Sync Class
      Base DN: dc=example,dc=com
      Filters: (objectClass=inetOrgPerson)
      DN Map: **,ou=users,dc=example,dc=com ->{1},CN=Users,DC=adsync,
       DC=PingIdentity,DC=com
      Synchronized Attributes: Custom set of mappings are defined
      Operations: Creates,Deletes,Modifies
```

11. To apply the configuration to the local PingDataSync Server instance, type **yes**. The configuration is recorded at `<server-root>/logs/tools/createsync-pipe-`

config.log.

Configure password encryption

This procedure is required if synchronizing passwords from an PingDirectory Server to Active Directory, or if synchronizing clear text passwords. These steps are not required if synchronizing from Active Directory to an PingData PingDirectory Server, or if not synchronizing passwords.

1. On the PingDirectory Server that will receive the password modifications, enable the Change Log Password Encryption component. The component intercepts password modifications, encrypts the password and adds an encrypted attribute, `ds-changelog-encrypted-password`, to the change log entry. The encryption key can be copied from the output if displayed, or accessed from the `<serverroot>/bin/sync-pipe-cfg.txt` file.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Changelog Password
Encryption" \
  --set enabled:true \
  --set changelog-password-encryption-key:<key>
```

2. On the PingDataSync Server, set the decryption key used to decrypt the user password value in the change log entries. The key allows the user password to be synchronized to other servers that do not use the same password storage scheme.

```
$ bin/dsconfig set-global-sync-configuration-prop \
  --set changelog-password-decryption-key:ej5u9e39pqo68
```

Test the configuration or populate data in the destination servers using bulk resync mode. See Using the resync Tool on the Identity Sync Server. Then, use `realtime-sync` to start synchronizing the data. See Using the realtime-sync Tool for more information. If synchronizing passwords, install the Password Sync Agent (PSA) on all of the domain controllers in the topology.

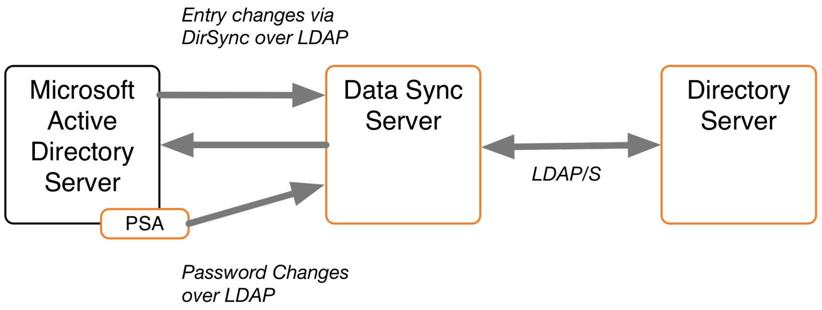
The Password Sync Agent

When synchronizing passwords with Active Directory systems, the PingDataSync Server requires that the Ping Identity Password Sync Agent (PSA) be installed on all domain controllers in the synchronization topology. This component provides real-time outbound password synchronization from Microsoft Active Directory to any supported Sync Destinations.

The PSA component provides password synchronization between directories that support differing password storage schemes. The PSA immediately hashes the password with a 160-bit salted secure hash algorithm and erases the memory where the clear-text password was stored. The component only transmits data over a secure (SSL) connection, and follows Microsoft's security guidelines when handling clear-text passwords. The PSA also uses Microsoft Windows password filters, which are part of the local security authority (LSA) process. The password filters enable implementing password policy validation and change notification mechanisms. For more information, see Microsoft's product documentation.

Note

For outbound password synchronization from an PingDirectory Server to Active Directory, enable the Password Encryption component. See Configuring the Password Encryption Component for more information.



Password Synchronization with Microsoft Active Directory

The PSA supports failover between servers. It caches the hashed password changes in a local database until it can be guaranteed that all PingDataSync Servers in the topology have received them. The failover features enable any or all of the PingDataSync Servers to be taken offline without losing any password changes from Active Directory.

The PSA is safe to leave running on a domain controller indefinitely. To stop synchronizing passwords, remove the `userPassword` attribute mapping in the PingDataSync Server, or stop the server. The PSA will not allow its local cache of password changes to grow out of control; it automatically cleans out records from its local database as soon as they have been acknowledged. It also purges changes that have been in the database for more than a week.

Before installing the PSA, consider the following:

- Make sure that the Active Directory domain controller has SSL enabled and running.
- Make sure the PingDataSync Servers are configured to accept SSL connections when communicating with the Active Directory host.
- At least one Active Directory Sync Source (ADSyncSource) needs to be configured on the PingDataSync Server and should point to the domain controller(s) on which the PSA will reside.
- At the time of installation, all PingDataSync Servers in the sync topology must be online and available.
- The PSA component is for outbound-only password synchronization from the Active Directory Systems. It is not necessary if performing a one-way password synchronization from the PingDirectory Server to the Active Directory server.

Install the Password Sync Agent

The PingDirectory Server distributes the PSA in zip file format with each PingDataSync Server package. The initial installation of the PSA requires a system restart.

Perform the following steps to install the PSA

1. On the domain controller, double-click the `setup.exe` file to start the installation.
2. Select a folder for the PSA binaries, local database, and log files.
3. Enter the host names (or IP addresses) and SSL ports of the PingDataSync Servers, such as `sync.host.com:636`. Do not add any prefixes to the hostnames.
4. Enter the Directory Manager DN and password. This creates an ADSync user on the PingDataSync Server.
5. Enter a secret key (password) that the sync agent uses when the ADSync user connects to the PingDataSync Server instances.
6. Click **Next** to begin the installation. All of the specified PingDataSync Servers are contacted, and any failures will roll back the installation. If everything succeeds, a message displays indicating that a restart is required. The PSA will start when the computer restarts, and the LSA process is loaded into memory. The LSA process cannot be restarted at runtime.
7. If synchronizing pre-encoded passwords from Active Directory to a Ping Identity system, allow pre-encoded passwords in the default password policy.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set allow-pre-encoded-passwords:true
```

Upgrade or Uninstall the Password Agent

The PingDataSync Server provides the `update` tool for upgrades to the server code, including the PSA. The upgrade does not require a restart, because the core password filter is already running under LSA. The upgrade replaces the implementation binaries, which are encapsulated from the password filter DLL.

To uninstall the PSA on the Active Directory system, use **Add/Remove Programs** on the Windows Control Panel. The implementation DLL will be unloaded, and the database and log files are deleted. Only the binaries remain.

The core password filter will still be running under the LSA process. It imposes zero overhead on the domain controller, because the implementation DLL has been unloaded. To remove the password filter itself (located at `C:\WINDOWS\System32\ubidPassFilt.dll`), restart the computer. On restart, the password filter and implementation binaries (in the install folder) can be deleted.

Note
The PSA cannot be reinstalled without another reboot.

Manually Configure the Password Sync Agent

Configuration settings for the Password Sync Service are stored in the Windows registry in `HKLM\SOFTWARE\UnboundID\PasswordSync`. Configuration values under this registry key can be modified during runtime. The agent automatically reloads and refresh its settings from the registry. Verify that the agent is working by checking the current log file, located in `<server-root>\logs\password-sync-current.log`.

Chapter 6: Synchronize with relational databases

The PingDataSync Server supports high-scale, highly-available data synchronization between the directory servers and relational database management systems (RDBMS). Any database with a JDBC 3.0 or later driver can be used.

Topics include:

Use the Server SDK

Synchronizing LDAP data to or from a relational database requires creating a JDBC Sync Source or Destination extension to act as an interface between the PingDataSync Server and the relational database. The Server SDK provides APIs to develop plug-ins and third-party extensions to the server using Java or Groovy. The Server SDK's documentation is delivered with the Server SDK build in zip format.

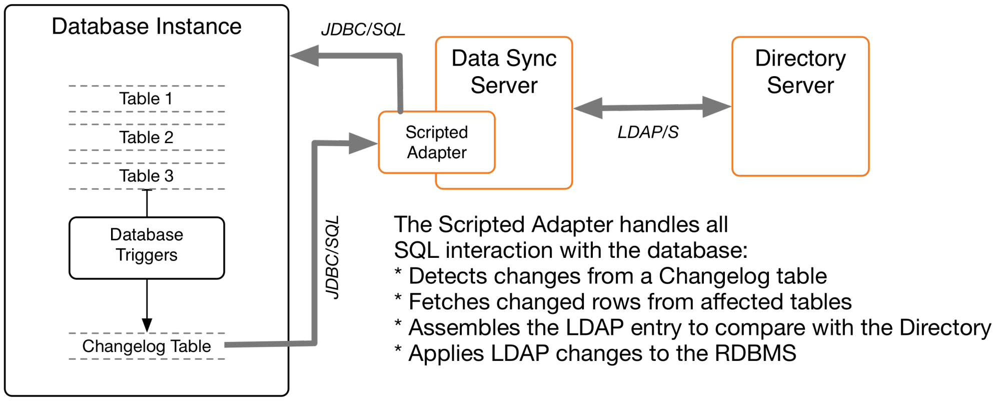
> **Note**
> Server SDK support is provided with Premium Support for the each product. Ping Identity does not provide support for the third party extensions developed using the Server SDK. Contact a Ping Identity support representative for assistance.

The Server SDK contains two abstract classes that correspond to how the database is used:

- `com.unboundid.directory.sdk.sync.api.JDBCSyncSource`
- `com.unboundid.directory.sdk.sync.api.JDBCSyncDestination`

The remainder of the SDK contains helper classes and utility functions to facilitate the script implementation. The SDK can use any change tracking mechanism to detect changes in the database. Examples are provided in the `<server-root>/config/jdbc/samples` directory for Oracle Database and Microsoft SQL Server.

The PingDataSync Server uses a scripted adapter layer to convert any database change to an equivalent LDAP entry. The Sync Pipe then processes the data through inclusive (or exclusive) filtering using attribute and DN maps defined in the Sync Classes to update the endpoint servers. For example, a script using Java can be configured by setting the `extension-class` property on a `ThirdPartyJDBCSyncSource` or `ThirdPartyJDBCSyncDestination` configuration object within the PingDataSync Server. The following is a sample architecture.


Database Instance
Table 1
Table 2
Table 3
Database Triggers
Changelog Table
JDBC/SQL
JDBC/SQL
Data Sync Server
Scripted Adapter
LDAP/S
Directory Server
The Scripted Adapter handles all
SQL interaction with the database:
* Detects changes from a Changelog table
* Fetches changed rows from affected tables
* Assembles the LDAP entry to compare with the Directory
* Applies LDAP changes to the RDBMS

Synchronizing with RDBMS Overview

The RDBMS synchronization process

The PingDataSync Server synchronizes data between a directory server and an RDBMS system with a Server SDK extension. The PingDataSync Server provides multiple configuration options, such as advanced filtering (fractional and subtree), attribute and DN mappings, transformations, correlations, and configurable logging.

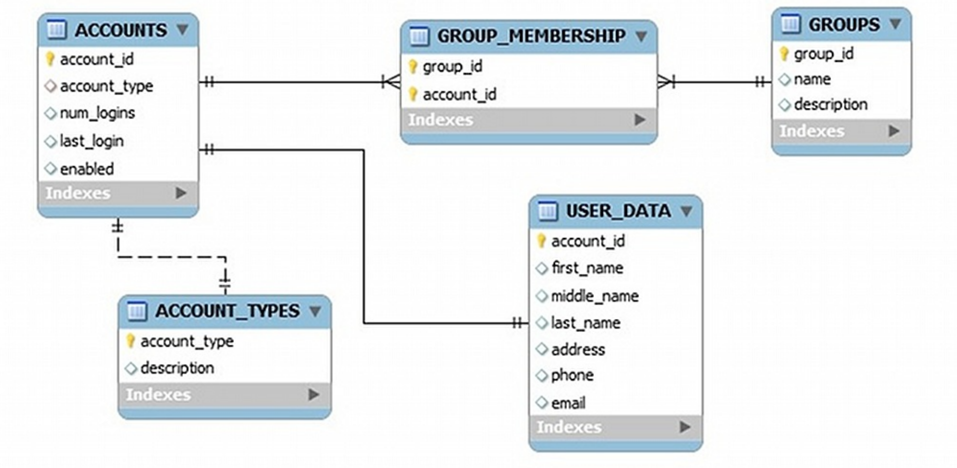
To support synchronizing changes, the database must be configured with a change tracking mechanism. An approach involving triggers, (one trigger per table) to record all changes to a change log table, is recommended. The database change log table Ping Identityshould record the type of change (INSERT, UPDATE, DELETE), the specific table name, the unique identifier for the changed row, the database entry type, the changed columns (from the source table), the modifier's name, and the change timestamp.

The PingDataSync Server delegates the physical interaction with the database to a user-defined extension, which has full control of the SQL queries. The extension layer provides flexibility in how the mapping semantics between the LDAP environment and the relational database environment are defined. The connection management, pooling, retry logic, and other boilerplate code are handled internally by the PingDataSync Server.

The RDBMS Synchronization (DBSync) implementation does not support failover between different physical database servers. Most enterprise databases have a built-in failover layer from which the PingDataSync Server can point to a single virtual address and port and still be highly available. A single RDBMS node can scale to multiple directory server endpoints.

DBSync example

ThePingDataSync Server provides a DBSync example between two endpoints consisting of an Ping Identity PingDirectory Server source and a RDBMS system, which will be used in this chapter. The entity-relational diagram for the normalized database schema is available in `<server-root>/config/jdbc/samples/oracle-db/ComplexSchema.jpg`, and is illustrated here:



Entry Relation Diagrams for the Schema Tables

Five tables are represented with their primary keys in bold. The entity relations and foreign keys are marked by the relationship lines. The illustration shows mapping to a custom object class on the directory server, while the "groups" table maps to a standard LDAP group entry with `objectclass:groupOfUniqueNames`.

Example directory server entries

The following example assumes that the directory server's schema is configured to handle the mapped attributes. If configuring a database-to-directory Sync Pipe with a newly installed directory server, make sure that the schema has the correct `attributeType` and `objectClass` definitions in place. The definitions can be added in a custom `99-user.ldif` file in the `config/schema` folder of the directory server, if necessary. The following are the LDAP entries that are used in the synchronization example:

```
dn: accountid=0,ou=People,dc=example,dc=com
objectClass: site-user
firstName: John
lastName: Smith
accountID: 1234
email: jsmith@example.com
phone: +1 556 805 4454
address: 17121 Green Street
numLogins: 4
lastLogin: 20070408182813.196Z
enabled: true

dn: cn=Group 1,ou=Groups,dc=example,dc=com
objectClass: groupOfUniqueNames
description: This is Group 1
uniqueMember: accountID=0,ou=People,dc=example,dc=com
uniqueMember: accountID=1,ou=People,dc=example,dc=com
```

Configure DBSync

Configuring a DBSync system includes extra tasks to create the extensions and to configure the database. The overall configuration process is as follows:

1. Download the appropriate JDBC driver to the PingDataSync Server's `/PingDataSync/lib` directory, and restart the server for the driver to load into the runtime.
2. Open the `java.properties` file with a text editor and add the `jdbc.drivers` argument. Save the file.
3. Run the `dsjavaproperties` command to apply the change. For example, enter the following for `start-sync-server`:

   ```
   start-sync-server.java-args=-d64 -server -Xmx256m -Xms256m -
   XX:+UseConcMarkSweepGC -
   ```

```
Djdbc.drivers=foo.bah.Driver:wombat.sql.Driver:com.example.OurDriver ...
etc.
```

4. Create one or more JDBC extensions based on the Server SDK. If configuring for bidirectional synchronization, two scripts are needed: one for the JDBC Sync Source; the other for the JDBC Sync Destination. Place the compiled extensions in the `/lib/extensions` directory.
5. Configure the database change log table and triggers (presented later). The vendor's native change tracking mechanism can be used, but a change log table should also be configured. Each table requires one database trigger to detect the changes and loads them into the change log table.
6. Configure the Sync Pipes, Sync Classes, external servers, DN and attribute maps for one direction.
7. Run the `resync --dry-run` command to test the configuration settings.
8. Run `realtime-sync set-startpoint` to initialize the starting point for synchronization.
9. Run the `resync` command to populate data on the destination endpoint.
10. Start the Sync Pipes using the `realtime-sync start` command.
11. Monitor the PingDataSync Server using the `status` commands and logs.
12. For bidirectional synchronization, configure another Sync Pipe, and repeat steps 4–8 to test the system.

Create the JDBC extension

The JDBC extension implementation must be written in Java, or the Groovy scripting language. Consult the Server SDK documentation for details on how to build and deploy extensions. The examples in this guide use Java. Java extensions are more strict and will catch programming errors during compile time rather than at runtime. Groovy is more flexible and can accomplish more with less lines of code.

Groovy scripts must reside in the `/lib/groovy-scripted-extensions` directory (Java implementations reside in `/lib/extensions`), which may also contain other plug-ins built using the Server SDK. If a script declares a package name, it must live under the corresponding folder hierarchy, just like a Java class. For example, to use a script class called `ComplexJDBCSyncSource` whose package is `com.unboundid.examples.oracle`, place it in `/lib/groovy-scripted-extensions/com/unboundid/examples/oracle` and set the `script-class` property on the Sync Source to `com.unboundid.examples.oracle.ComplexJDBCSyncSource`. There are a few reference implementations provided in the `config/jdbc/samples` directory. Use the `manage-extension` tool in the `bin` directory, or `bat` (Windows) to install or update the extension. See the Server SDK Extensions section for more information.

Note
Any changes to an existing script require a manual Sync Pipe restart. Any configuration change automatically restarts the affected Sync Pipe.

The default libraries available on the classpath to the script implementation include:

- Groovy
- LDAP SDK for Java
- JRE

Logging from within a script can be done with the Server SDK's `ServerContext` abstract class. Some of `ServerContext` methods are not available when the `resync` tool runs, because it runs outside of the PingDataSync Server Server process. Any logging during a `resync` operation is saved to the `logs/tools/resync.log` file.

Implement a JDBC Sync Source

The `JDBCSyncSource` abstract class must be implemented to synchronize data from a relational database. Since the PingDataSync Server is LDAP-centric, this class enables database content to be converted into LDAP entries. For more detailed information on the class, see the Server SDK Javadoc.

The extension imports classes from the Java API, LDAP SDK for Java API, and the Server SDK. Depending on the data, implement the following methods:

- `initializeJDBCSyncSource` – Called when a Sync Pipe first starts, or when the `resync` process starts. Any initialization should be performed here, such as creating internal data structures and setting up variables.
- `finalizeJDBCSyncSource` – Called when a Sync Pipe stops, or when the `resync` process stops. Any clean up should be performed here, and all internal resources should be freed.
- `setStartpoint` – Sets the starting point for synchronization by identifying the starting point in the change log. This method should cause all changes previous to the specified start point to be disregarded and only changes after that point to be returned by the `getNextBatchOfChanges` method. There are several different startpoint types (see SetStartpointOptions in the Server SDK), and this implementation is not required to support them all. If the specified startpoint type is unsupported, this method throws an exception (`IllegalArgumentException`). This method can be called from two different contexts: when the `realtime-sync set-startpoint` command is used (the Sync Pipe is required to be stopped) or immediately after a connection is established to the source server.

Note
The `RESUME_AT_SERIALIZABLE` startpoint type must be supported. This method is used when a Sync Pipe first starts and loads its state from disk.

- `getStartpoint` – Gets the current value of the startpoint for change detection.
- `fetchEntry` – Returns a full source entry (in LDAP form) from the database, corresponding to the `DatabaseChangeRecord` object that is passed. The `resync` command also uses this class to retrieve entries.
- `acknowledgeCompletedOps` – Provides a means for the PingDataSync Server to acknowledge to the database which operations have completed processing.

Note
The internal value for the startpoint should only be updated after a synchronization operation is acknowledged in to this script (through this method). If not, changes could be missed when the PingDataSync Server is restarted.

- `getNextBatchOfChanges` – Retrieves the next set of changes for processing. The method also provides a generic means to limit the size of the result set.
- `listAllEntries` – Used by the `resync` command to get a listing of all entries.
- `cleanupChangelog` – In general, we recommend implementing a `cleanupChangelog` method, so that the PingDataSync Server can purge old records from the change log table, based on a configurable age.

See the `config/jdbc/samples` directory for example script implementations and the Server SDK Javadoc for more detailed information on each method.

Implement a JDBC Sync Destination

The `JDBCSyncDestination` abstract class must be implemented to synchronize data into a relational database. The class enables converting LDAP content to database content. The extension imports classes from the Java API, LDAP SDK for Java API, and the Server SDK, depending on the database configuration. Implement the following methods in the script:

- `initializeJDBCSyncDestination` – Called when a Sync Pipe starts, or when the `resync` process starts. Any initialization should be performed here, such as creating internal data structures and setting up variables.
- `finalizeJDBCSyncDestination` – Called when a Sync Pipe stops, or when the `resync` process stops. Any clean up should be performed here, and all internal resources should be freed.
- `createEntry` – Creates a full database entry (or row), corresponding to the LDAP entry that is passed in.
- `modifyEntry` – Modifies a database entry, corresponding to the LDAP entry that is passed in.
- `fetchEntry` – Returns a full destination database entry (in LDAP form), corresponding to the source entry that is passed in.

- `deleteEntry` – Deletes a full entry from the database, corresponding to the LDAP entry that is passed in.

For more detailed information on the abstract class, consult the Server SDK Javadoc.

Configure the database for synchronization

Configuring the database for synchronization includes defining:

- a database SyncUser account
- the change tracking mechanism
- the database triggers (one per table) for the application

The following procedure uses the example setup script in `/config/jdbc/samples/oracle-db/OracleSyncSetup.sql`. Items in brackets are user-named labels.

Note
Database change tracking necessary if synchronizing FROM the database. If synchronizing TO a database, configure the Sync User account and the correct privileges.

1. Create an Oracle login (`SyncUser`) for the PingDataSync Server, so that it can access the database server. Grant sufficient privileges to the `SyncUser` for any tables to be synchronized, and change the default password.

```
CREATE USER SyncUser IDENTIFIED BY password
DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;
GRANT "RESOURCE" TO SyncUser;
GRANT "CONNECT" TO SyncUser;
```

2. Create change log tables on the database as follows:

```
CREATE TABLE ubid_changelog (
  --This is the unique number for the change change_number Number NOT NULL
PRIMARY KEY,
  --This is the type of change (insert, update, delete). NOTE: This should
represent
  --the actual type of change that needs to happen on the destination(for
example a
  --database delete might translate to a LDAPmodify, etc.)
    change_type VARCHAR2(10) NOT NULL,

  --This is the name of the table that was changed table_name VARCHAR(50)
NOT NULL,
  --This is the unique identifier for the row that was changed. It is up
to
  --the trigger code to construct this, but it should follow a DN-like
format
  --(e.g. accountID={accountID}) where at least the primary key(s) are
  --present. If multiple primary keys are required, they should be
delimited
  --with a unique string, such as '%%' (e.g. accountID={accountID}%%
  --groupID={groupID})
    identifier VARCHAR2(100) NOT NULL,

  --This is the database entry type. The allowable values for this must be
  --set on the JDBC Sync Source configuration within the Synchronization
  --Server.
    entry_type VARCHAR2(50) NOT NULL,

  --This is a comma-separated list of columns from the source table that
were updated as part of
  --this change.
    changed_columns VARCHAR2(1000) NULL,

  --This is the name of the database user who made the change
    modifiers_name VARCHAR2(50) NOT NULL,

  --This is the timestamp of the change
    change_time TIMESTAMP(3) NOT NULL, CONSTRAINT chk_change_type
    CHECK (change_type IN ('insert','update','delete'))) ORGANIZATION
INDEX;
```

3. Create an Oracle function to get the `SyncUser` name. This is a convenience function for the triggers.

```
CREATE OR REPLACE FUNCTION get_sync_user RETURN VARCHAR2
IS
BEGIN
  RETURN 'SyncUser';
END get_sync_user;
```

4. Create an Oracle sequence object for the change-number column in the change log table.

```
CREATE SEQUENCE ubid_changelog_seq MINVALUE 1 START WITH 1
NOMAXVALUE INCREMENT BY 1 CACHE 100 NOCYCLE;
```

5. Create a database trigger for each table that will participate in synchronization. An example, located in `/config/jdbc/samples/oracle-db/OracleSyncSetup.sql`, shows a trigger for the Accounts table that tracks all changed columns after any INSERT, UPDATE, and DELETE operation. The code generates a list of changed items and then inserts them into the change log table.

Considerations for synchronizing to database destination

When configuring a directory-to-database Sync Pipe, the following are recommended:

- **Identify the Object Classes** – Create a Sync Class per object class, so that they can easily be distinguished and have different mappings and synchronization rules.

- For each Sync Class, set the following items in the configuration menus using the `dsconfig` tool.
    - **Set the Include-Filter Property** – Make sure the `include-filter` property is set on the Sync Class configuration menu to something that will uniquely identify the source entries, such as `objectClass=customer`.
    - **Create Specific Attribute Mappings** – Create an attribute mapping for every LDAP attribute to be synchronized to a database column, add these to a single attribute map, and set it on the Sync Class. With this configured, the script does not need to know about the schema on the directory side. A constructed attribute mapping that maps a literal value to the `objectClass` attribute can be added to the script to determine the database entry type. For example, `"account" -> objectClass` can be added, which would result in the constructed destination LDAP entry always containing an `objectClass` of `"account."` If needed, a mulit-valued `conditional-value-pattern` property can be used to conditionalize the attribute mapping based on the subtype of the entry or on the value of the attribute. See Conditional Value Mapping for additional information.
    - **Create Specific DN Maps (optional)** – If necessary, create a DN map that recognizes the DN's of the source entries and maps them to a desired destination DN. In most cases, the script will use the attributes rather than the DN to figure out which database entry needs to be changed.
    - **Set auto-mapped-source-attribute to "-none-"** – Remove the default value of "-all-" from the `auto-mapped-source-attribute` on the Sync Class configuration menu, and replace it with "-none-."
- **Configure Create-Only Attributes** – Any attributes that should be included when created, but never modified (such as `objectclass`) should be specified on the Sync Pipe as a `create-only` attribute. If the PingDataSync Server ever computes a difference in that attribute between the source and destination, it will not try to modify it at the destination. To avoid bidirectional loop-back, set the `ignore-changes-by-[user|dn]` property on both Sync Sources when configuring for bidirectional synchronization.
- **Synchronizing DELETE Operations** – On PingDirectory Server and Nokia 8661 Directory Server systems, configure the `changelog-deleted-entry-include-attribute` property on the changelog backend menu using the `dsconfig` tool. This property allows for the proper synchronization of DELETE operations. For more information, see Configuring the Directory Server Backend for Synchronizing Deletes.
- **Attribute-Synchronization-Mode for DBSync** – For MODIFY operations, the PingDataSync Server detects any change on the source change log, fetches the source entry, applies mappings, computes the equivalent destination entry, fetches the actual destination entry, and then runs a diff between the two entries to determine the minimal set of changes to synchronize. By default, changes on the destination entry are made

only for those attributes that were detected in the original change log entry. This is configurable using the `attribute-synchronization-mode` property, which sets the type of diff operation that is performed between the source and destination entries.

If the source endpoint is a database server, set the `attribute-synchronization-mode` property to `all-attributes` on the Sync Class configuration menu. The diff operation will consider all source attributes. Any that have changed will be updated on the destination, even if the change was not originally detected in the change log. This mode is useful when a list of changed columns in the database may not be available. If both endpoints are directory servers, use the default configuration of `modified-attributes-only` to avoid possible replication conflicts.

- **Handling MODDN Operations** – The concept of renaming an entry (modifyDN) does not have a direct equivalent for relational databases. The `JDBCSyncDestination` API does not handle changes of this type. Instead, the `modifyEntry()` method is called as if it is a normal change. The extension can verify if the entry was renamed by looking at the `SyncOperation` that is passed in (`syncOperation.isModifyDN()`). If true, the `fetchedDestEntry` parameter will have the old DN. The new DN can be obtained by calling `syncOperation.getDestinationEntryAfterChange()`.

Configure a directory-to-database Sync Pipe

The following configures a one-way Sync Pipe with an PingDirectory Server as the Sync Source and an RDBMS (Oracle) system as the Sync Destination with the `create-sync-pipe-config` tool. Sync Pipes can be configured later using `dsconfig`.

Create the Sync Pipe

The following procedures configure the Sync Pipe, external servers, and Sync Classes. The examples are based on the Complex JDBC sample in the `config/jdbc/samples/oracle-db` directory.

The `create-sync-pipe-config` tool can be run with the server offline and the configuration can later be imported.

1. Run the `create-sync-pipe-config` tool.

   ```
   $ bin/create-sync-pipe-config
   ```

2. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.
3. On the Synchronization Mode menu, select Standard Mode or Notification Mode.
4. On the Synchronization Directory menu, choose one-way or bidirectional synchronization.

Configure the Sync Source

1. On the Source Endpoint Type menu, enter the number for the sync source corresponding to the type of source external server.
2. Enter a name for the Source Endpoint.
3. Enter the base DN for the directory server, which is used as the base for LDAP searches. For example, enter `dc=example,dc=com`, and then press **Enter** again to return to the menu. If entering more than one base DN, make sure the DNs do not overlap.
4. On the Server Security menu, select the type of communication that the PingDataSync Server will use with the endpoint servers.
5. Enter the host and port of the source endpoint server. The Sync Source can specify a single server or multiple servers in a replicated topology. The server tests that a connection can be established.
6. Enter the DN of the Sync User account and create a password for this account. The Sync User account enables the PingDataSync Server to access the source endpoint server. By default, the Sync User account is stored as `cn=SyncUser,cn=Root DNs,cn=config`.

Configure the destination endpoint server

1. On the Destination Endpoint Type menu, select the type of data store on the endpoint server. This example is configuring an Oracle Database.
2. Enter a name for the Destination Endpoint.
3. On the JDBC Endpoint Connection Parameters menu, enter the fully-qualified host name or IP address for the Oracle database server.
4. Enter the listener port for the database server, or press **Enter** to accept the default (1521).
5. Enter a database name such as `dbsync-test`.
6. The server attempts to locate the JDBC driver in the `lib` directory. If the file is found, a success message is displayed.

   ```
   Successfully found and loaded JDBC driver for:
   jdbc:oracle:thin:@//dbsync-w2k8-vm-2:1521/dbsync-test
   ```

   If the server cannot find the JDBC driver, add it later, or quit the `create-sync-pipe-config` tool and add the file to the `lib` directory.
7. Add any additional JDBC connection properties for the database server, or press **Enter** to accept the default (no). Consult the JDBC driver’s vendor documentation for supported properties.

8. Enter a name for the database user account with which the PingDataSync Server will communicate, or press **Enter** to accept the default (SyncUser). Enter the password for the account.
9. On the Standard Setup menu, enter the number for the language (Java or Groovy) that was used to write the server extension.
10. Enter the fully qualified name of the Server SDK extension class that implements the JDBCSyncDestination API.

```
Enter the fully qualified name of the Java class that will implement
com.unboundid.directory.sdk.sync.api.JDBCSyncDestination:
com.unboundid.examples.oracle.ComplexJDBCSyncDestination
```

11. Configure any user-defined arguments needed by the server extension. These are defined in the extension itself and the values are specified in the server configuration. If there are user-defined arguments, enter **yes**.
12. To prepare the Source Endpoint server, which tests the connection to the server with the Sync User account, press **Enter** to accept the default (yes). For the Sync User account, it will return "Denied" as the account has not been written yet to the Directory Server at this time.

```
Testing connection to server1.example.com:1389 ..... Done
Testing 'cn=Sync User,cn=Root DNs,cn=config' access ..... Denied
```

13. To configure the Sync User account on the directory server, press **Enter** to accept the default (yes). Enter the bind DN (`cn=Directory Manager`) and the bind DN password of the directory server so that you can configure the `cn=Sync User` account. The PingDataSync Server creates the Sync User account, tests the base DN, and enables the change log.

```
Created 'cn=Sync User,cn=Root DNs,cn=config'
Verifying base DN 'dc=example,dc=com' ..... Done
Enabling cn=changelog .....
```

14. Enter the maximum age of the change log entries, or press **Enter** to accept the default.

Configure the Sync Pipe and Sync Classes

The following procedures define a Sync Pipe and two Sync Classes. The first Sync Class is used to match the `accounts` objects. The second Sync Class matches the `group` objects.

1. Continuing from the previous session, enter a name for the Sync Pipe.
2. When prompted to define one or more Sync Classes, enter **yes**.

Configure the accounts Sync Class

1. Enter a name for the Sync Class. For example, type `accounts_sync_class`.
2. If restricting entries to specific subtrees, enter one or more base DNs. If not, press **Enter** to accept the default (no).
3. To set an LDAP search filter, type **yes** and enter the filter "`(accountid=*)`". Press **Enter** again to continue. This property sets the LDAP filters and returns all entries that match the search criteria to be included in the Sync Class. In this example, specify that any entry with an `accountID` attribute be included in the Sync Class. If the entry does not contain any of these values, it will not be synchronized to the target server.
4. Choose to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization, or press **Enter** to accept the default (all).
5. Specify the operations that will be synchronized for the Sync Class, or press **Enter** to accept the default.

Configure the groups Sync Class

For this example, configure another Sync Class to handle the `groups` objectclass. The procedures are similar to that of the configuration steps for the `account_sync_class` Sync Class.

1. On the Sync Class menu, enter a name for a new sync class, such as `groups_sync_class`.
2. To restrict entries to specific subtrees, enter one or more base DNs.
3. Set an LDAP search filter. Type **yes** to set up a filter and enter the filter "`(objectClass=groupOfUniqueNames)`." This property sets the LDAP filters and returns all entries that match the `groupOfUniqueNames` attribute to be included in the Sync Class. If the entry does not contain any of these values, it will not be synchronized to the target server.
4. Choose to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization, or press **Enter** to accept the default (all).
5. Specify the operations that will be synchronized for the Sync Class, or press **Enter** to accept the default.
6. At the prompt to enter the name of another Sync Class, press **Enter** to continue.
7. On the Default Sync Class Operations menu, press **Enter** to accept the default. The Default Sync Class determines how all entries that do not match any other Sync Class are handled.
8. Review the configuration, and press **Enter** to write the configuration to the server.

Use the `dsconfig` tool to make changes to this configuration. See Configuring the PingDataSync Server for configuration options and details.

Considerations for synchronizing from a database source

When synchronizing from a database to a directory or RDBMS server, the following are recommended:

- **Identify Database Entry Types** – Identify the database entry types that will be synchronized, and:
  - Set the `database-entry-type` property on the JDBC Sync Source (this is required), and make sure the entry types are what the triggers are inserting into the change tracking mechanism.
  - Create a Sync Class per entry type, and set different mappings and rules for each one.
- For each Sync Class, do the following:
  - Make sure the `include-filter` property is set to match the entry type.
  - Create a specific attribute mapping for every database column to be synchronized to an LDAP attribute and set it on the Sync Class. If this is done, the script will not have to know about the schema on the directory side.
  - Create a DN map that recognizes the DNs generated by the script and maps them to the correct location at the destination.
  - Remove the default value of "-all-" from the `auto-mapped-source-attribute` property on the Sync Class, and replace it with the value `objectClass`. The object class for the fetched source entry is determined by the scripted layer. Values from the database should not be automatically mapped to an attribute with the same name, except the `objectclass` attribute, which should map directly for CREATE operations. If this is not done, an error is generated.
  - Change the `destination-correlation-attributes` property to contain the attributes that uniquely represent the database entries on the directory server destination.
- **Avoid Bidirectional Loopback** – Set the `ignore-changes-by-[user|dn]` property on both Sync Sources when configuring for bidirectional synchronization, to make sure that changes are not looped back by the PingDataSync Server.

See Use the create-sync-pipe tool to configure synchronization for details about creating the Sync Pipe.

Synchronize a specific list of database elements

The `resync` command enables synchronizing a specific set of database keys that are read from a JDBC Sync Source file using the `--sourceInputFile` option. The contents of the file are passed line-by-line into the `listAllEntries()` method of the `JDBCSyncSource` extension, which is used for the Sync Pipe. The method processes the input and returns `DatabaseChangeRecord` instances based on the input from the file.

Perform the following steps to synchronize a specific list of database elements using the `resync` tool:

1. Create a file of JDBC Sync Source elements. There is no set format for the file, but it typically contains a list of primary keys or SQL queries. For example, create a file containing a list of primary keys and save it as `sourceSQL.txt`.

   ```
   user.0
   user.1
   user.2
   user.3
   ```

2. Run the `resync` command with the `--sourceInputFile` option to run on individual primary keys in the file.

   ```
   $ bin/resync --pipe-name "dbsync-pipe" \
     --sourceInputFile sourceSQL.txt
   ```

3. If searching for a specific type of database entry, use the `--entryType` option that matches one of the configured entry types in the JDBCSyncSource.

   ```
   $ bin/resync --pipe-name "dbsync-pipe" \
     --entryType account \
     --sourceInputFile sourceSQL.txt
   ```

Chapter 7: Synchronize through PingDirectoryProxy Servers

Because most data centers deploy directory servers in a proxied environment, the PingDataSync Server can also synchronize data through a proxy server in both load-balanced and entry-balancing deployments. The following proxy endpoints are supported:

- Ping Identity PingDirectoryProxy Servers
- Nokia 8661 Directory Proxy Servers

This chapter details a Sync-through-Proxy deployment and provides background information on how it works. Before setting up the PingDataSync Server, review the *Ping Identity PingDirectoryProxy Server Administration Guide* for information about the PingDirectoryProxy Server.

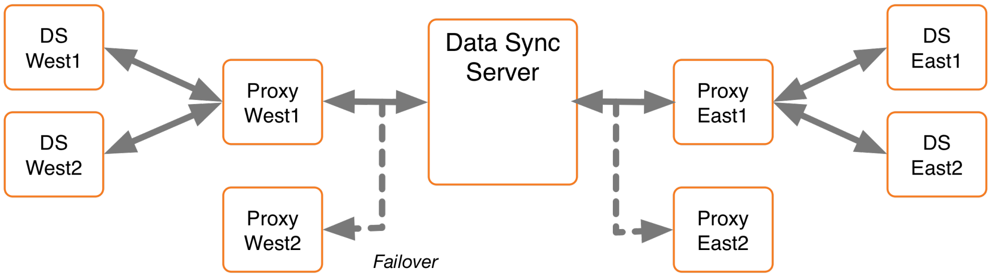
Topics include:

Synchronization through a Proxy Server overview

To handle data synchronization through a proxy server, PingData servers have a `cn=changelog` state management system that supports a token-based API. In a standard, non-proxied configuration, the PingDataSync Server polls the source server for changes, determines if a change is necessary, and fetches the full entry from the source. Then, it finds the corresponding entry in the destination endpoint using correlation rules and applies the minimal set of changes. The server fetches and compares the full entries to make sure it does not synchronize any stale data from the change log.

In a proxied environment, the PingDataSync Server passes the request through a proxy server to the backend set of directory servers. The PingDataSync Server uses the highest priority proxy server designated in its endpoint server configuration and can use others in the event of a failover.

The following illustrates a deployment with two endpoints consisting of a proxy server deployment in front of the backend set of directory servers.



Synchronization Through Proxy Example

Change log operations

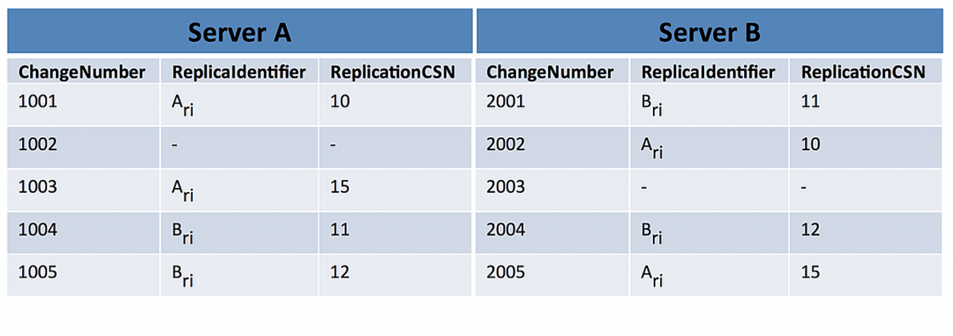
When the PingDataSync Server runs a poll for any changes, it sends a get change log batch extended request to the `cn=changelog` backend. The batch request looks for entries in the change log and asks for the server ID, change number, and replica state for each change. The PingDirectoryProxy Server routes the request to a directory server instance, which then returns a changed entry plus a token identifying the server ID, change number and replica state for each change. The PingDirectoryProxy Server then sends a get change log batch response back to the PingDataSync Server with this information. For entry-balancing deployments, the PingDirectoryProxy Server must "re-package" the directory server tokens into its own proxy token to identify the specific data set.

The first time that the PingDataSync Server issues the batch request, it also issues a get server ID request to identify the specific server ID that is processing the extended request. The PingDirectoryProxy Server routes the request to the directory server instance, and then returns a server ID in the response. With the next request, the PingDataSync Server sends a 'route to server' request that specifies the server instance to access again in this batch session. It also issues a server ID request in the event that the particular server is down. This method avoids round-robin server selection and provides more efficient overall change processing.

PingDirectory Server and PingDirectoryProxy Server tokens

The PingDirectory Server maintains a change log database index to determine when to resume sending changes (for ADD, MODIFY, or DELETE operations) in its change log. While a simple stand-alone directory server can track its resume point by the last change number sent, replicated servers or servers deployed in entry balancing environments have a different change number ordering in its change log because updates can come from a variety of sources.

The following illustrates two change logs in two replicated directory servers, server A and B. "A" represents the replica identifier for a replicated subtree in Server A, and "B" represents the replica identifier for the same replicated subtree in server B. The replica identifiers with a hyphen ("-") mark any local, non-replicated but different changes. While the two replicas record all of the changes, the two change logs have two different change number orderings because updates come in at different times.



Server A			Server B		
ChangeNumber	ReplicaIdentifier	ReplicationCSN	ChangeNumber	ReplicaIdentifier	ReplicationCSN
1001	A_{ri}	10	2001	B_{ri}	11
1002	-	-	2002	A_{ri}	10
1003	A_{ri}	15	2003	-	-
1004	B_{ri}	11	2004	B_{ri}	12
1005	B_{ri}	12	2005	A_{ri}	15

Different Change Number Order in Two Replicated Change Logs

To track the change log resume position, the PingDirectory Server uses a change log database index to identify the latest change number position corresponding to the highest replication CSN number for a given replica. This information is encapsulated in a directory server token and returned in the get change log batch response to the PingDirectoryProxy Server. The token has the following format:

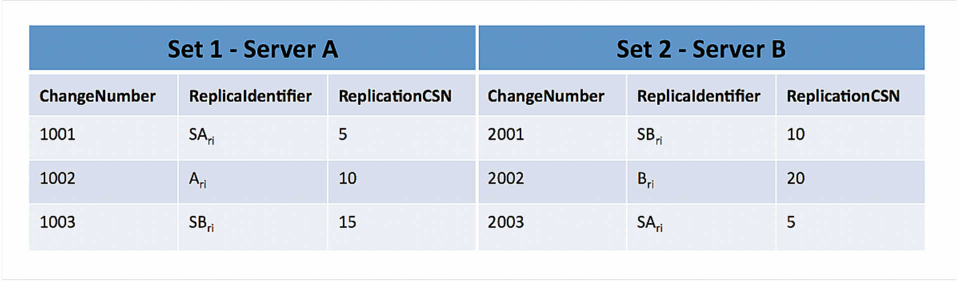
```
Directory Server Token: server ID, changeNumber, replicaState
```

For example, if the PingDirectoryProxy Server sends a request for any changed entries, and the directory servers return the change number 1003 from server A and change number 2005 from server B, then each directory server token would contain the following information:

```
Directory Server Token A:
  serverID A, changeNumber 1003, replicaState {15(A)}

Directory Server Token B:
  serverID B, changeNumber 2005, replicaState {12(B), 15(A)}
```

Change log tracking in entry balancing deployments

Change log tracking can become more complex in that a shared area of data can exist above the entry-balancing base DN in addition to each backend set having its own set of changes and tokens. In the following figure, Server A belongs to an entry-balancing set 1, and server B belonging to an entry-balancing set 2. Shared areas that exist above the entry-balancing base DN are assumed to be replicated to all servers. "SA" represents the replica identifier for that shared area on Server A and "SB" represents the replica identifier for the same area on Server B.



Set 1 - Server A			Set 2 - Server B		
ChangeNumber	ReplicaIdentifier	ReplicationCSN	ChangeNumber	ReplicaIdentifier	ReplicationCSN
1001	SA_{ri}	5	2001	SB_{ri}	10
1002	A_{ri}	10	2002	B_{ri}	20
1003	SB_{ri}	15	2003	SA_{ri}	5

Different Change Number Order in Two Replicated Change Logs

The PingDirectoryProxy Server cannot pass a directory server token from the client to the directory server and back again. In an entry-balancing deployment, the PingDirectoryProxy Server must maintain its own token mechanism that associates a directory server token (`changeNumber`, `replicaIdentifier`, `replicaState`) to a particular backend set.

```
Proxy Token:
backendSetID 1: ds-token 1 (changeNumber, replicaIdentifier, replicaState)
backendSetID 2: ds-token 2 (changeNumber, replicaIdentifier, replicaState)
```

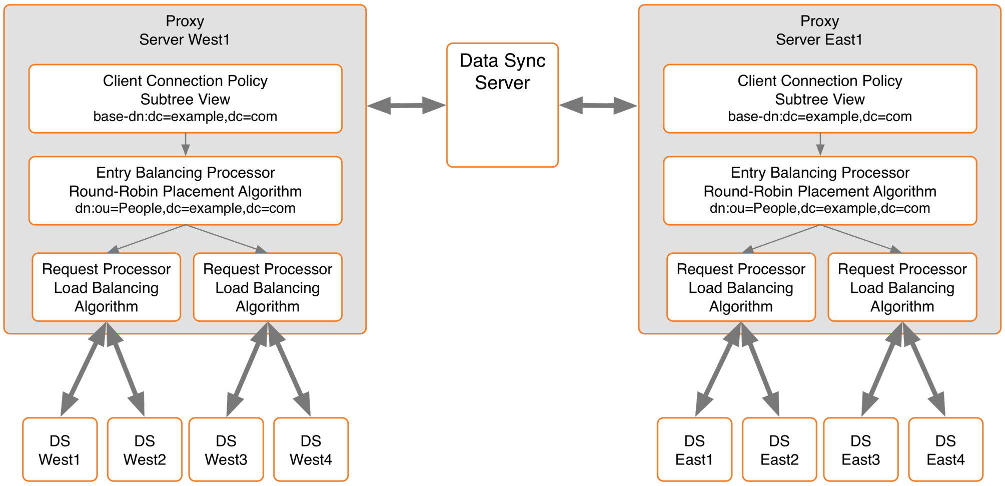
For example, if the PingDirectoryProxy Server returned change 1002 from Server A and change 2002 from Server B, then the Proxy token would contain the following:

```
Proxy Token:
backendSetID 1: ds-token-1 {serverID A, changeNumber 1002, replicaState (5
(SA), 15(A)}
backendSetID 2: ds-token-2 {serverID B, changeNumber 2002, replicaState (10
(SB), 20(B)}
```

For each change entry returned by a backend, the PingDirectoryProxy Server must also decide whether it is a duplicate of a change made to the backend set above the entry-balancing base. If the change is a duplicate, then it is discarded. Otherwise, any new change is returned with a new value of the proxy token.

Example configuration

This following configures synchronization through a proxy and use two endpoints consisting of a PingDirectoryProxy Server with a backend set of PingDirectory Servers: both sets are replicated. The PingDirectoryProxy Server uses an entry-balancing environment for the DN `ou=People,dc=example,dc=com` and provides a subtree view for `dc=example,dc=com` in its client connection policy. For this example, communication is over standard LDAP and failover servers are not installed or designated in the PingDataSync Server.



Example Synchronization Through Proxy Configuration

Configure the source PingDirectory Server

The following procedures configure a backend set of directory servers. The procedure is the same for the source servers and the destination servers in a synchronization topology. For directory server installation and configuration details, see the *Ping Identity PingDirectory Server Administration Guide*.

1. On each backend Directory Server that will participate in synchronization, enable the change log database, either from the command line or by using a `dsconfig` batch file.

```
$ dsconfig --no-prompt set-backend-prop \
  --backend-name changelog \
  --set enabled:true
```

2. Stop the server if it is running, and import the dataset for the first backend set into the first server in the backend set prior to the import.

```
$ bin/stop-server
```

```
$ bin/import-ldif --backendID userRoot --ldifFile ../dataset.ldif
```

```
$ bin/start-server
```

3. On the first server instance in the first backend set, configure replication between this server and the second server in the same backend set.

```
$ bin/dsreplication enable --host1 ldap-west-01.example.com \
  --port1 389 \
  --bindDN1 "cn=Directory Manager" \
  --bindPassword1 password \
  --replicationPort1 8989 \
  --host2 ldap-west-02.example.com \
  --port2 389 \
  --bindDN2 "cn=Directory Manager" \
  --bindPassword2 password \
  --replicationPort2 9989 \
  --adminUID admin \
  --adminPassword admin \
  --baseDN dc=example,dc=com \
  --no-prompt
```

4. Initialize the second server in the backend set with data from the first server in the backend set. This command can be run from either instance.

```
$ bin/dsreplication initialize \
  --hostSource ldap-west-01.example.com \
  --portSource 389 \
  --hostDestination ldap-west-02.example.com \
  --portDestination 389 \
  --baseDN "dc=example,dc=com" \
  --adminUID admin \
  --adminPassword admin \
  --no-prompt
```

5. Run the following command to check replica status.

```
$ bin/dsreplication status \
  --hostname ldap-west-01.example.com \
  --port 389 \
  --adminPassword admin \
  --no-prompt
```

6. Repeat steps 3 through 6 (import, enable replication, initialize replication, check status) for the second backend set.

Configure a Proxy Server

The following procedures configure a proxy server, including defining the external servers and configuring the client-connection policy. The procedure is the same for the source servers and the destination servers in a synchronization topology. For additional changes, use the `dsconfig` tool. For proxy installation and configuration details, see the *Ping Identity PingDirectoryProxy Server Administration Guide*.

1. From the PingDirectoryProxy Server root directory, run the `prepare-external-server` command to set up the `cn=Proxy User` account for access to the backend directory servers. The server tests the connection and creates the `cn=Proxy User` account.

```
$ bin/prepare-external-server --no-prompt \
  --hostname ldap-west-01.example.com \
  --port 389 --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --proxyBindDN "cn=Proxy User,cn=Root DNs,cn=config" \
  --proxyBindPassword pass \
  --baseDN "dc=example,dc=com"
```

2. Repeat step 1 for any other directory server instances.
3. Run the `dsconfig` command to define the external servers and their types. For this example, round-robin load balancing algorithms are defined, which do not require health checks or locations to be specified.

```
$ bin/dsconfig --no-prompt create-external-server \
  --server-name ldap-west-01 \
  --type "ping-identity-ds" \
  --set "server-host-name:ldap-west-01.example.com" \
  --set "server-port:389" \
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-external-server \
  --server-name ldap-west-02 \
  --type "ping-identity-ds" \
  --set "server-host-name:ldap-west-02.example.com" \
  --set "server-port:389" \
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-external-server \
  --server-name ldap-west-03 \
  --type "ping-identity-ds" \
  --set "server-host-name:ldap-west-03.example.com" \
  --set "server-port:389" \
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-external-server
  --server-name ldap-west-04 \
  --type "ping-identity-ds" \
  --set "server-host-name:ldap-west-04.example.com" \
  --set "server-port:389" \
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

4. Create a load-balancing algorithm for each backend set.

```
$ bin/dsconfig --no-prompt create-load-balancing-algorithm \
  --algorithm-name "test-lba-1" \
  --type "round-robin" --set "enabled:true" \
  --set "backend-server:ldap-west-01" \
  --set "backend-server:ldap-west-02" \
  --set "use-location:false" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-load-balancing-algorithm \
  --algorithm-name "test-lba-2" \
  --type "round-robin" --set "enabled:true" \
  --set "backend-server:ldap-west-03"
  --set "backend-server:ldap-west-04"
  --set "use-location:false" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

5. Configure the proxying request processors, one for each load-balanced directory server set. A request processor provides the logic to either process the operation directly, forward the request to another server, or hand off the request to another request processor.

```
$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "proxying-processor-1" --type "proxying" \
  --set "load-balancing-algorithm:test-lba-1" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "proxying-processor-2" --type "proxying" \
  --set "load-balancing-algorithm:test-lba-2" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

6. Define an entry-balancing request processor. This request processor is used to distribute entries under a common parent entry among multiple backend sets. A backend set is a collection of replicated directory servers that contain identical portions of the data. Multiple proxying request processors are used to process operations.

   Next, define the placement algorithm, which selects the server set to use for new add operations to create new entries. In this example, a round-robin placement algorithm forwards LDAP add requests to backends sets.

```
$ bin/dsconfig --no-prompt create-placement-algorithm \
  --processor-name "entry-balancing-processor" \
  --algorithm-name "round-robin-placement" \
  --set "enabled:true" \
  --type "round-robin" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

7. Define the subtree view that specifies the base DN for the entire deployment.

```
$ bin/dsconfig --no-prompt create-subtree-view \
  --view-name "test-view" \
  --set "base-dn:dc=example,dc=com" \
  --set "request-processor: entry-balancing-processor" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

8. Finally, define a client connection policy that specifies how the client connects to the proxy server.

```
$ bin/dsconfig --no-prompt set-client-connection-policy-prop \
  --policy-name "default" \
  --add "subtree-view:test-view" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

Configuring the PingDataSync Server

Configure the PingDataSync Server once the PingDirectoryProxy Server and its backend set of PingDirectory Server instances are configured and fully functional for each endpoint, which are labeled as `ldap-west` and `ldap-east` in this example. For information on installing and configuring the PingDataSync Server, see Installing the PingDataSync Server.

1. From the PingDataSync Server root directory, run the `create-sync-pipe-config` tool.

```
$ bin/create-sync-pipe-config
```

2. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.
3. On the Synchronization Mode menu, press **Enter** to select Standard mode.
4. On the Synchronization Directory menu, choose the option for one-way or bidirectional synchronization.
5. On the First Endpoint Type menu, enter the number for the type of backend data store for the first endpoint. In this example, type the number corresponding to the PingDirectoryProxy Server.

```
>>>> First Endpoint Type
Enter the type of data store for the first endpoint:
1) Ping Identity Directory Server
2) Ping Identity Directory Proxy Server
3) Alcatel-Lucent Directory Server
4) Alcatel-Lucent Proxy Server
5) Sun Directory Server
```

```
6) Microsoft Active Directory
7) Microsoft SQL Server
8) Oracle Database
9) Custom JDBC

b) back
q) quit

Enter choice [1]: 2
```

6. Enter a descriptive name for the first endpoint.
7. Enter the base DN where the PingDataSync Server can search for the entries on the first endpoint server.
8. Specify the type of security when communicating with the endpoint server.
9. Enter the hostname and port of the endpoint server. The PingDataSync Server tests the connection. Repeat this step if configuring another server for failover.
10. Enter the Sync User account that will be used to access the endpoint server, or press **Enter** to accept the default `cn=Sync User,cn=Root DNs,cn=config`. Enter a password for the account.
11. The first endpoint deployment is defined using the PingDirectoryProxy Server (`ldap-west`). Repeat steps 5-10 to define the second proxy deployment (`ldap-east`) on the PingDataSync Server.
12. Prepare the endpoint servers in the topology. This step confirms that the Sync User account is present on each server and can communicate between the PingDataSync Server and the PingDirectoryProxy Servers. In addition to preparing the PingDirectoryProxy Server, the PingDataSync Server prepares the backend set of directory servers as the proxy server passes through the authorization to access these servers.
13. Repeat the previous step to prepare the second endpoint server. If other servers have not been prepared, make sure that they are prior to synchronization.
14. Define the Sync Pipe from proxy 1 to proxy 2. In this example, accept the default "Ping Identity Proxy 1 to Ping Identity Proxy 2."
15. To customize on a per-entry basis how attributes get synchronized, define one or more Sync Classes. Create a Sync Class for the special cases, and use the default Sync Class for all other mappings.
16. For the default Sync Class Operations, specify the operations that will be synchronized.
17. Review the configuration settings, and write the configuration to the PingDataSync Server in the `sync-pipe-cfg.txt` file.

Test the configuration

If the `create-sync-pipe-config` tool was not used to create the synchronization configuration, two properties must be verified on each endpoint: `proxy-server` and `use-changelog-batch-request`. The `proxy-server` property should specify the name of the proxy server. The `use-changelog-batch-request` should be set to `true` on the Sync Source only. The `use-changelog-batch-request` is not available on the destination endpoint.

The PingDataSync Server connection parameters (hostname, port, bind DN, and bind password) are required.

1. The following commands check the properties on a Sync Source.

   On the Sync Source:

   ```
   $ bin/dsconfig --no-prompt \
     get-sync-source-prop \
     --source-name "Ping Identity Proxy 1" \
     --property "proxy-server" \
     --property "use-changelog-batch-request"
   ```

   On the Sync Destination:

   ```
   $ bin/dsconfig --no-prompt \
     get-sync-source-prop \
     --source-name "Ping Identity Proxy 2" \
     --property "proxy-server"
   ```

2. From the server root directory, run the `dsconfig` command to set a flag indicating that the endpoints are PingDirectoryProxy Servers:

   ```
   $ bin/dsconfig --no-prompt \
     set-sync-source-prop \
     --source-name "Ping Identity Proxy 1" \
     --set proxy-server:ldap-west-01 \
     --set use-changelog-batch-request:true
   ```

   ```
   $ bin/dsconfig --no-prompt \
     set-sync-source-prop \
     --source-name "Ping Identity Proxy 2" \
     --set proxy-server:ldap-east-01
   ```

3. Run the `resync --dry-run` command to test the configuration settings for each Sync Pipe and debug any issues.

   ```
   $ bin/resync --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
   --dry-run
   ```

4. Run `realtime-sync set-startpoint` to initialize the starting point for synchronization.

   ```
   $ realtime-sync set-startpoint --end-of-changelog \
     --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2" \
     --port 389 \
     --bindDN "cn=Directory Manager" \
     --bindPassword password
   ```

Note
For synchronization through proxy deployments, the `--change-number` option cannot be used with the `realtime-sync set-startpoint` command, because the PingDataSync Server cannot retrieve specific change numbers from the backend directory servers. Use the `--change-sequence-number`, `--end-of-changelog`, or other options available for the tool.

5. Run the `resync` command to populate data on the endpoint destination server if necessary.

```
$ bin/resync --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
\
  --numPasses 3
```

6. Start the Sync Pipe using the `realtime-sync start` command.

```
$ bin/realtime-sync start \
  --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
```

7. Monitor the PingDataSync Server using the status commands and logs.

Index the LDAP changelog

The PingData PingDirectory Server and the Nokia 8661 Directory Server (3.0 or later) both support attribute indexing in the changelog backend to enable get changelog batch requests to filter results that include only changes of specific attributes. For example, in an entry balanced proxy deployment, the PingDataSync Server sends a get changelog batch request to the Proxy Server, which will send out individual requests to each backend server.

Each directory server that receives a request must iterate over the whole range of changelog entries, and then match entries based on search criteria for inclusion in the batch. The majority of this processing involves determining whether a changelog entry includes changes to a particular attribute or set of attributes, or not. Changelog indexing can dramatically speed up throughput when targeting specific attributes.

Attribute indexing is configured using the `index-include-attribute` and `index-exclude-attribute` properties on the changelog backend. The properties can accept the specific attribute name or special LDAP values "*" to specify all user attributes or "+" to specify all operational attributes.

Perform the following steps to configure changelog indexing:

1. On all source directory servers, enable changelog indexing for the attributes that will be synchronized. Use the `index-include-attribute` and `index-exclude-attribute` properties. The following example specifies that all user attributes (`index-include-attribute:*`) be indexed in the changelog, except the description and location attributes (`index-exclude-attribute:description` and `index-exclude-attribute:location`).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "index-include-attribute:*" \
  --set "index-exclude-attribute:description \
  --set "index-exclude-attribute:location
```

Note
There is little performance and disk consumption penalty when using `index-include-attribute:*` with a combination of `index-exclude-attribute` properties, instead of explicitly defining each attribute using `index-include-attribute`. The only cautionary note about using `index-include-attribute:*` is to be careful that unnecessary attributes get indexed.

2. On the PingDataSync Server, configure the `auto-map-source-attributes` property to specify the mappings for the attributes that need to be synchronized.

The PingDataSync Server will write a NOTICE message to the error log when the Sync Pipe first starts, indicating whether the server is using changelog indexing or not.

```
[30/Mar/2016:13:21:36.781 -0500] category=SYNC severity=NOTICE
msgID=1894187256 msg="Sync Pipe 'TestPipe' is not using changelog indexing on
the source server"
```

Changelog synchronization considerations

If the Sync Source is configured with `use-changelog-batch-request=true`, the PingDataSync Server will use the get changelog batch request to retrieve changes from the LDAP changelog. This extended request can contain an optional set of selection criteria, which specifies changelog entries for a specific set of attributes.

The PingDataSync Server takes the union of the source attributes from DN mappings, attribute mappings, and the `auto-mapped-source-attributes` property on the Sync Class to create the selection criteria. However, if it encounters the value "-all-" in the `auto-mapped-source-attributes` property, it cannot make use of selection criteria because the Sync Pipe is interested in all possible source attributes.

When the PingDirectory Server receives a get changelog request that contains selection criteria, it returns changelog entries for one or more of the attributes that meet the criteria.

- For ADD and MODIFY changelog entries, the changes must include at least one attribute from the selection criteria.
- For MODDN changelog entries, one of the RDN attributes must match the selection criteria.
- For DELETE changelog entries, one of the `deletedEntryAttrs` much match the selection criteria.

If `auto-mapped` is not set to `all` source attributes, at least one should be configured to show up in the `deletedEntryAttrs` (with the `changelog-deleted-entry-include-attribute` property on the changelog backend).

Another way to do this is to set `use-reversible-form` to `true` on the changelog backend. This includes all attributes in the `deletedEntryAttrs`.

Chapter 8: Synchronize in Notification Mode

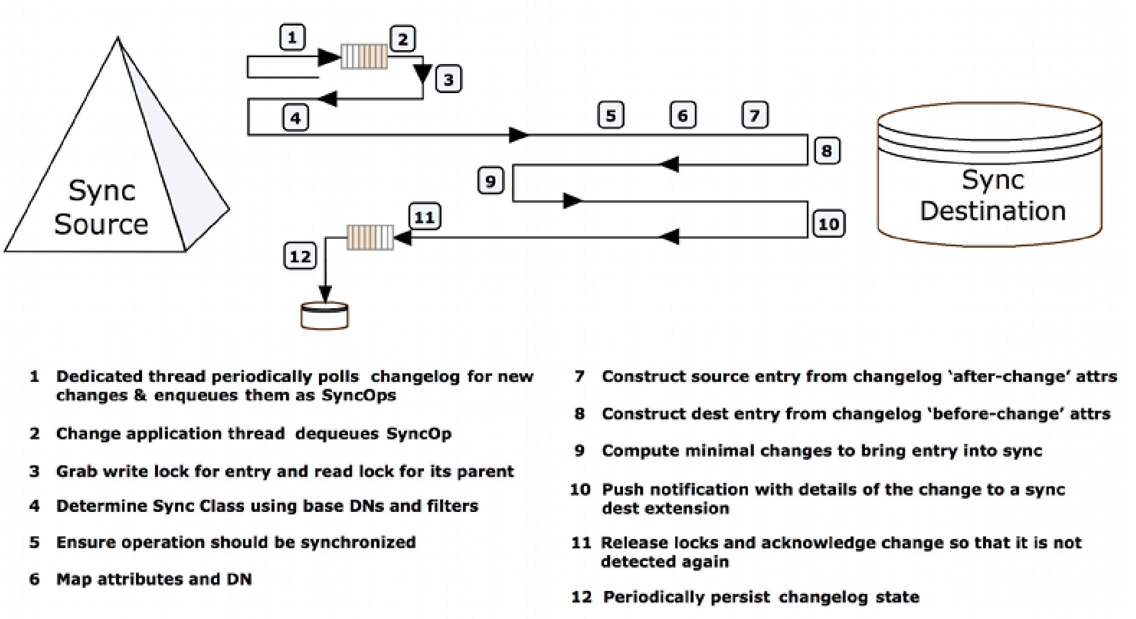
The PingDataSync Server supports a notification synchronization mode that transmits change notifications on a source endpoint to third-party destination applications. As with standard mode, notifications can be filtered based on the type of entry that was changed, the specific attributes that were changed, and the type of change (ADD, MODIFY, DELETE). The PingDataSync Server can send a notification to arbitrary endpoints by using a custom server extension.

Topics include:

Notification mode overview

The PingDataSync Server supports standard and notification synchronization modes. Notification Mode polls the directory server’s LDAP change log for changes on any entry but skips the fetch and compare phases of processing of Standard Mode. Instead, the Sync Destination is notified of the change regardless of the current state of that entry at the source or destination. The PingDataSync Server accesses state information on the change log to reconstruct the before-and-after values of any modified attribute (for example, for MODIFY change operation types). It passes in the change information to a custom server extension based on the Server SDK.

Third-party libraries can be employed to customize the notification message to an output format required by the client application or service. For example, the server extension can use a third-party XML parsing library to convert the change notifications to a SOAP XML format. Notification mode can only be used with an PingDirectory Server, Nokia 8661 Directory Server, PingDirectoryProxy Server, or Nokia 8661 Directory Proxy Server as the source endpoint.



1 Dedicated thread periodically polls changelog for new changes & enqueues them as SyncOps

2 Change application thread dequeues SyncOp

3 Grab write lock for entry and read lock for its parent

4 Determine Sync Class using base DNs and filters

5 Ensure operation should be synchronized

6 Map attributes and DN

7 Construct source entry from changelog ‘after-change’ attrs

8 Construct dest entry from changelog ‘before-change’ attrs

9 Compute minimal changes to bring entry into sync

10 Push notification with details of the change to a sync dest extension

11 Release locks and acknowledge change so that it is not detected again

12 Periodically persist changelog state

Notification Mode Synchronization Change Flow

The PingDataSync Server can use notification mode with any type of endpoint; therefore, it is not an absolute requirement to have a custom server extension in your system. For example, it is possible to set up a notification Sync Pipe between two LDAP server endpoints.

Implementation Considerations

Before implementing and configuring a Sync Pipe in notification mode, answer the following questions:

- What is the interface to client applications?
- What type of connection logic is required?
- How will the custom server extension handle timeouts and connection failures?
- What are the failover scenarios?
- What data needs to be included in the change log?
- How long do the change log entries need to be available?
- What are the scalability requirements for the system?
- What attributes should be used for correlation?
- What should happen with each type of change?
- What mappings must be implemented?

Use the Server SDK and LDAP SDK

To support notification mode, the Server SDK provides a `SyncDestination` extension to synchronize with any client application. The PingDataSync Server engine processes the notification and makes it available to the extension, which can be written in Java or Groovy. This generic extension type can also be used for standard synchronization mode.

Similar to database synchronization, the custom server extension is stored in the `<server-root>/lib/groovy-scripted-extensions` folder (for Groovy-based extensions) or the jar file in the `<server-root>/lib/extensions` folder (for Java-based extensions) prior to configuring the PingDataSync Server for notification mode. Groovy scripts are compiled and loaded at runtime.

The Server SDK's `SyncOperation` interface represents a single synchronized change from the Sync Source to the Sync Destination. The same `SyncOperation` object exists from the time a change is detected, through when the change is applied at the destination.

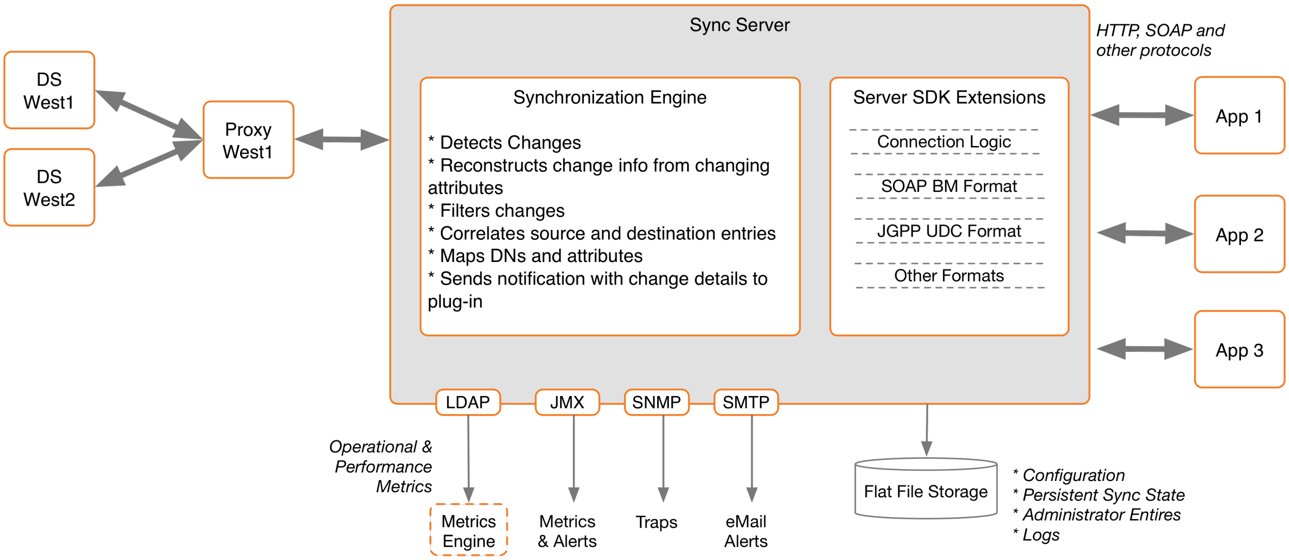
The LDAP SDK's `UnboundIDChangelogEntry` class (in the `com.unboundid.ldap.sdk.unboundidds` package) has high level methods to work with the `ds-changelog-before-value`, `ds-changelogafter-values`, and `ds-changelog-entry-key-attr-values` attributes. The class is part of the commercial edition of the LDAP SDK for Java and is installed automatically with the PingDataSync Server. For detailed information and examples, see the LDAP SDK Javadoc.

Notification mode architecture

Notification mode, a configuration setting on the Sync Pipe, requires a one-way directional Sync Pipe from a source endpoint topology to a target client application. The PingDataSync Server detects the changes in the PingDirectory Server’s LDAP change log, filters the results specified in the Sync Classes, applies any DN and attribute mappings, then reconstructs the change information from the change log attributes. A server extension picks up the notification arguments from the `SyncOperation` interface (part of the Server SDK) and converts the data to the desired output format. The server extension establishes the connections and protocol logic to push the notification information to the client applications or services. All of the operations, administration, and management functions available in standard mode, such as monitoring, (LDAP, JMX, SNMP), alerts (JMX, SNMP, SMTP), and logging features are the same for notification mode.

> **Note**
> The Server SDK includes documentation and examples on how to create a directory server extension to support notification mode.

For a given entry, the PingDataSync Server sends notifications in the order that the changes occurred in the change log even if a modified attribute has been overwritten by a later change. For example, if an entry's `telephoneNumber` attribute is changed three times, three notifications will be sent in the order they appeared in the change log.



Notification Mode Architecture

Sync Source requirements

A separate Sync Pipe is required for each client application that should receive a notification. The Sync Sources must consist of one or more instances of the following directory or proxy servers with the PingDataSync Server:

- Ping IdentityPingDirectory Server and PingDirectoryProxy Server (version 3.0.5 or later)
- Nokia 8661 Directory Server
- The Sync Destination can be of any type

> **Note** While the PingDirectoryProxy Server and Nokia 8661 Directory Proxy Server can front other vendor's directory servers, such as Active Directory and Sun DSEE, for processing LDAP operations, the PingDataSync Server cannot synchronize changes from these sources through a proxy server.

Synchronizing changes directly from Active Directory and Sun DSEE cannot be done with notification mode.

Failover Capabilities

For sync source failovers, configure replication between the Directory Servers to ensure data consistency between the servers. A PingDirectoryProxy Server can also front the backend PingDirectory Server set to redirect traffic, if connection to the primary server fails. A PingDirectoryProxy Server must be used for synchronizing changes in an entry-balancing environment. Once the primary PingDirectory Server is online, it assumes control with no information loss as its state information is kept across the backend PingDirectory Servers.

For sync destination failovers, connection retry logic must be implemented in the server extension, which will use the Sync Pipe's advanced property settings to retry failed operations. There is a difference between a connection retry and an operation retry. An extension should not retry operations because the PingDataSync Server does this automatically. However, the server extension is responsible for re-establishing connections to a destination that has gone down, or failing over to an alternate server. The server extension can also be designed to trigger its own error- handling code during a failed operation.

For PingDataSync Server failovers, the secondary PingDataSync Servers will be at or slightly behind the state where the primary server initiated a failover. Both primary and secondary PingDataSync Servers track the last failed acknowledgement, so once the primary server fails over to a secondary server, the secondary server will not miss a change.

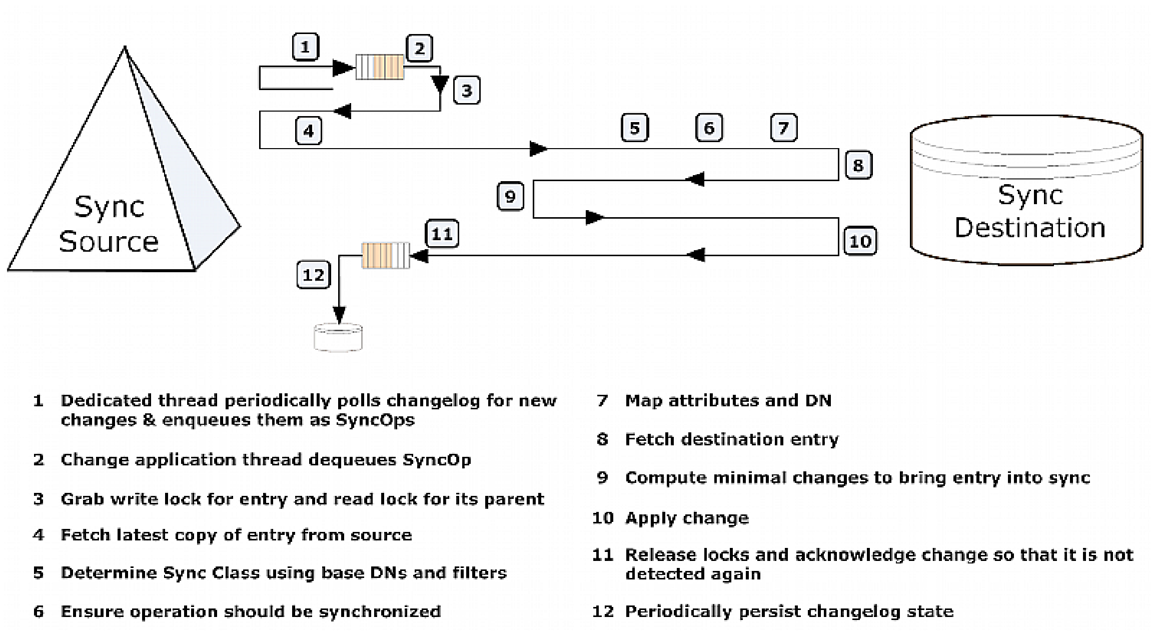
Note
If failover is a concern between PingDataSync Servers, change the `sync-failover-polling-interval` property from 7500 ms to a smaller value. This will result in a quicker failover, but will marginly increase traffic between the two PingDataSync Servers. The `sync-failover-connection-timeout` and `sync-failover-response-timeout` properties may also be updated to use different failover timeout durations. Use `dsconfig` to access the property on the Global Sync Configuration menu.

Notification Sync Pipe change flow

Multi-threaded Sync Pipes allow the PingDataSync Server to process multiple notifications in parallel in the same manner as synchronizing changes in standard mode, which increases throughput and offsets network latency. A single change-detection thread pulls in batches of change log entries and queues them internally. To guarantee consistency, the PingDataSync Server's internal locking mechanisms ensure the following properties:

- Changes to the same entry will be processed in the same order that they appear in the change log.
- Changes to parent entries will be processed before changes to its children.
- Changes to entries with the same RDN value are handled sequentially.

The number of concurrent threads is configurable on the Sync Pipe using the `num-worker-threads` property. In general, single-threading should be avoided.



1 Dedicated thread periodically polls changelog for new changes & enqueues them as SyncOps

2 Change application thread dequeues SyncOp

3 Grab write lock for entry and read lock for its parent

4 Fetch latest copy of entry from source

5 Determine Sync Class using base DNs and filters

6 Ensure operation should be synchronized

7 Map attributes and DN

8 Fetch destination entry

9 Compute minimal changes to bring entry into sync

10 Apply change

11 Release locks and acknowledge change so that it is not detected again

12 Periodically persist changelog state

Notification Sync Pipe Change Flow

Configure Notification mode

The PingDataSync Server supports notification mode with the following components:

Use the create-sync-pipe-config tool

The `create-sync-pipe-config` tool supports the configuration of notification mode. Any pre-existing Sync Sources can be read from the local configuration (in the `config.ldif` file).

No resync command functionality

The `resync` function is disabled on a Sync Pipe in notification mode as its functionality is not supported in this implementation. Notification mode views the directory server's change log as a rolling set of data that pushes out change notifications to its target application.

LDAP change log features required for notifications

The PingDirectory Server and the Nokia 8661 Directory Server require the following advanced global change log properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

These properties are enabled and configured during the `create-sync-pipe-config` configuration process on the PingDataSync Server. The properties can also be enabled on the directory servers using the `dsconfig` advanced properties setting on the Backend Changelog menu.

changelog-include-key-attribute

The changelog-include-key-attribute property specifies one or more attributes that should always be included in the change log entry. These are attributes needed to correlate entries between the source and destination, such as uid, employeeNumber, or mail. These properties are also needed for evaluating any filters in the Sync Class. For example, if notifications are only sent for user entries, and the Sync Class included the filter (objectclass=people), the objectclass attribute must be configured as a changelog-include-key-attribute so that the Sync Pipe can evaluate the inclusion criteria when processing the change. In standard mode, values needed in the filter are read from the entry itself after it is fetched instead of from the changelog entry. These attributes are always included in a change log entry, also called a change record, regardless if they have changed or not.

The changelog-include-key-attribute property causes the current (after-change) value of the specified attributes to be recorded in the ds-changelog-entry-key-attr-values attribute on the change log entry. This applies for all change types. During a delete operation, the values are from the entry before it was deleted. The key values are recorded on every change and override any settings configured in the changelog-include-attribute, changelog-exclude-attribute, changelog-deleted-entry-include-attribute, or changelog-deleted-entry-exclude-attribute properties in the directory server changelog (see the *Ping Identity PingDirectory Server Configuration Reference* for more information).

Normal LDAP to LDAP synchronization topologies typically use dn as a correlation attribute. If dn is used as a correlation attribute only, the changelog-include-key-attribute property does not need to be set. However, if another attribute is used for correlation, this property must be set during the Sync Pipe configuration.

The LDAP change log attribute, ds-changelog-entry-key-attr-values, stores the attribute that is always included in a change log entry on every change for correlation purposes. In addition to regular attributes, virtual and operational attributes can be specified as entry keys.

To view an example, see the *Ping Identity PingDirectory Server Administration Guide*.

changelog-max-before-after-values

The changelog-max-before-after-values property specifies the maximum number of "before and after" values (default 200) that should be stored for any changed attribute in the change log. Also, when enabled, it will add the ds-changelog-before-values and ds-changelog-after-values attributes to any change record that contains changes (for Modify and ModifyDN).

The main purpose of the changelog-max-before-after-values property is to ensure that an excessively large number of changes is not stored for multi-valued attributes. In most cases, the directory server’s schema defines a multi-valued attribute to be unlimited in an entry. For example, if a group entry whose member attribute references 10000 entries, the desire may be to not have all of the attributes if a new member added.

If either the ds-changelog-before-values or the ds-changelog-after-values attributes exceed the count set in the changelog-max-before-after-values property, the attribute values are no longer stored in a change record but its attribute name and number is stored in the ds-changelog-attr-exceeded-max-values-count attribute, which appears in the change record.

In addition to this property, set the use-reversible-form property to TRUE. This guarantees that sufficient information is stored in the change log for all operation types to be able to replay the operations at the destination. The create-sync-pipe-config tool configures these properties when it prepares the servers.

The changelog-max-before-after-values property configures the following change log attributes:

- ds-changelog-before-values – Captures all "before" values of a changed attribute. It will store up to the specified value in the changelog-max-before-after-values property (default 200).
- ds-changelog-after-values – Captures all "after" values of a changed attribute. It will store up to the specified value in the changelog-max-before-after-values property (default 200).
- ds-changelog-attr-exceeded-max-values-count – Stores the attribute names and number of before and after values on the change log entry after the maximum number of values (set by the changelog-max-before-after-values property) has been exceeded. This is a multi-valued attribute with the following format:

  ```
  attr=attributeName,beforeCount=200,afterCount=201
  ```

  where attributeName is the name of the attribute and the beforeCount and afterCount are the total number of values for that attribute before and after the change, respectively. In either case (before or after the change), if the number of values is exceeding the maximum, those values will not be stored.

LDAP change log for Notification and Standard Mode

Both Notification and Standard mode Sync Pipes can consume the same LDAP Change Log without affecting the other. Standard mode polls the change record in the change log for any modifications, fetches the full entries on the source and the destination, and then compares them for the specific changes. Notification mode gets the before and after values of a changed attribute to reconstruct an entry, and bypasses the fetch-and-compare phase. Both can consume the same LDAP Change Log with no performance loss or conflicts.

Note
If the configuration obtains the change log through a PingDirectoryProxy Server, the contents of the change log will not change as it is being read from the change logs on the directory server backend.

Implementing the Server Extension

Notification mode relies heavily on the server extension code to process and transmit the change using the required protocol and data formats needed for the client applications. Create the extension using the Server SDK, which provides the APIs to develop code for any destination endpoint type. The Server SDK’s documentation (Javadoc and examples) is delivered with the Server SDK built-in zip format. The SDK provides all of the necessary classes to extend the functionality of the PingDataSync Server without code changes to the core product. Once the server extension is in place, use other third-party libraries to transform the notification to any desired output format.

Consider the following when implementing the extension:

- **Use the manage-extension Tool** – Use the `manage-extension` tool in the `bin` directory or `bat` directory (Windows) to install or update the extension. See Managing Extensions for more information.
- **Review the Server SDK Package** – Review Server SDK documentation and examples before building and deploy a Java or Groovy extension.
- **Connection and Protocol Logic** – The Server SDK extension must manage the notification connection and protocol logic to the client applications.
- **Implementing Extensions** – Test the create methods, the delete methods, and the modify methods for each entry type. Update the configuration as needed. Finally, package the extensions for deployment. Logging levels can be increased to include more details.
- **Use the SyncOperation Type** – The `SyncOperation` class encapsulates everything to do with a given change. Objects of this type are used in all of the synchronization SDK extensions. See the Server SDK Javadoc for the `SyncOperation` class for information on the full set of methods.
- **Use the EndpointException Type** – The Sync Destination type offers an exception type called `EndpointException` to extend a standard Java exception and provide custom exceptions. There is also logic to handle LDAP exceptions, using the LDAP SDK.
- **About the PostStep result codes** – The `EndpointException` class uses `PostStep` result codes that are returned in the server extension:
  - `retry_operation_limited` – Retry a failed attempt up to the limit set by `max_operation_attempts`.
  - `retry_operation_unlimited` – Retry the operation an unlimited number of times until a success, abort, or `retried_operation_limited`. This should only be used when the destination endpoint is unavailable.
  - `abort_operation` – Abort the current operation without any additional processing.
- **Use the ServerContext class for logging** – The `ServerContext` class provides several logging methods which can be used to generate log messages and/or alerts from the scripted layer: `logMessage()`, `sendAlert()`, `debugCaught()`, `debugError()`, `debugInfo()`, `debugThrown()`, `debugVerbose()`, and `debugWarning()`. These are described in the Server SDK API Javadocs. Logging related to an individual `SyncOperation` should be done with the `SyncOperation#logInfo` and `SyncOperation#logError` methods.

- **Diagnosing Script Errors** – When a Groovy extension does not behave as expected, first look in the error log for stack traces. If `classLoader` errors are present, the script could be in the wrong location or may not have the correct package. Groovy checks for errors at runtime. Business logic errors must be systematically found by testing each operation. Make sure logger levels are set high enough to debug.

Configuring the Notification Sync Pipe

The following procedure configures a one-way Sync Pipe with a PingDirectory Server as the Sync Source and a generic sync destination. The procedure uses the `create-sync-pipe-config` tool in interactive command-line mode and highlights the differences for configuring a Sync Pipe in notification mode.

Considerations for Configuring Sync Classes

When configuring a Sync Class for a Sync Pipe in notification mode, consider the following:

- Exclude any operational attributes from synchronizing to the destination so that its before and after values are not recorded in the change log. For example, the following attributes can be excluded: `creatorsName`, `createTimeStamp`, `ds-entry-unique-id`, `modifiersName`, and `modifyTimeStamp`. Filter the changes at the change log level instead of making the changes in the Sync Class to avoid extra configuration settings with the following:
    - Use the directory server's `changelog-exclude-attribute` property with (+) to exclude all operational attributes (`change-log-exclude-attribute:+`).
    - Configure a Sync Class that sets the `excluded-auto-mapped-source-attributes` property to each operational attribute to exclude from the synchronization process.
    - Use the directory server's `changelog-exclude-attribute` property to specify each operational attribute to exclude in the synchronization process. Set the configuration using the `dsconfig` tool on the directory server Change Log Backend menu. For example, `set changelog-exclude-attribute:modifiersName`.
- Use the `destination-create-only-attribute` advanced property on the Sync Class. This property sets the attributes to include on CREATE operations only.
- Use the `replace-all-attr-values` advanced property on the Sync Class. This property specifies whether to use the ADD and DELETE modification types (reversible), or the REPLACE modification type (non-reversible) for modifications to destination entries. If set to `true`, REPLACE is used.
- If targeting specific attributes that require higher performance throughput, consider implementing change log indexing. See Synchronizing Through Proxy Servers for more information.

Creating the Sync Pipe

The initial configuration steps show how to set up a single Sync Pipe from a directory server instance to a generic Sync Destination.

Before starting:

- Place any third-party libraries in the `<server-root>/lib/extensions` folder.
- Implement a server extension for any custom endpoints and place it in the appropriate directory.

1. If necessary, start the PingDataSync Server:

   ```
   $ bin/start-server
   ```

2. Run the `create-sync-pipe-config` tool.

   ```
   $ bin/create-sync-pipe-config
   ```

3. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.
4. On the Synchronization Mode menu, select the option for notification mode.
5. On the Synchronization Directory menu, enter the option to create a one-way Sync Pipe in notification mode from directory to a generic client application.

Configuring the Sync Source

1. On the Source Endpoint Type menu, enter the option for the Sync Source type.
2. Choose a pre-existing Sync Source, or create a new sync source.
3. Enter a name for the Source Endpoint and a name for the Sync Source.
4. Enter the base DN for the directory server used for LDAP searches, such as `dc=example,dc=com`, and press **Enter** to return to the menu. If entering more than one base DN, make sure they do not overlap.
5. On the Server Security menu, select the type of communication that the PingDataSync Server will use with endpoint servers.
6. Enter the host and port of the first Source Endpoint server. The Sync Source can specify a single server or multiple servers in a replicated topology. The PingDataSync Server contacts this first server if it is available, then contacts the next highest priority server if the first server is unavailable. The server tests the connection.
7. On the Sync User Account menu, enter the DN of the sync user account and password, or press **Enter** to accept the default, `cn=Sync User,cn=Root DNs,cn=config`. This account allows the PingDataSync Server to access the source endpoint server.

Configure the Destination Endpoint Server

1. On the Destination Endpoint Type menu, select the type of data store on the endpoint server. In this example, select the option for Custom.
2. Enter a name for the Destination Endpoint and a name for the Sync Destination.
3. On the Notifications Setup menu, select the language (Java or Groovy) used to write the server extension.
4. Enter the fully qualified name of the Server SDK extension that implements the abstract class. A Java, extension should reside in the `/lib/extensions` directory. A Groovy script should reside in the `/lib/groovy-scripted-extensions` directory.
5. Configure any user-defined arguments needed by the server extension. Typically, these are connection arguments, which are defined by the extension itself. The values are then entered here and stored in the server configuration.
6. Configure the maximum number of before and after values for all changed attributes. Notification mode requires this. Set the cap to something well above the maximum number of values that any synchronized attribute will have. If this cap is exceeded, the PingDataSync Server will issue an alert. For this example, we accept the default value of 200.

   ```
   Enter a value for the max changelog before/after values,
   or -1 for no limit [200]:
   ```

7. Configure any key attributes in the change log that should be included in every notification. These attributes can be used to find the destination entry corresponding to the source entry, and will be present whether or not the attributes changed. Later, any attributes used in a Sync Class include-filter should also be configured as key attributes in the Sync Class.
8. In both standard and notification modes, the Sync Pipe processes the changes concurrently with multiple threads. If changes must be applied strictly in order, the number of Sync Pipe worker threads will be reduced to 1. This will limit the maximum throughput of the Sync Pipe.

The rest of the configuration steps follow the same process as a standard synchronization mode Sync Pipe. See About the Sync User Account for more information.

Access control filtering on the Sync Pipe

The PingDataSync Server provides an advanced Sync Pipe configuration property, `filter-changes-by-user`, that performs access control filtering on a changelog entry for a specific user.

Since the changelog entry contains data from the target entry, the access controls filter out attributes that the user does not have the privileges to see before it is returned. For example, values in the changes, `ds-changelog-before-values`, `ds-changelog-after-values`, `ds-changelog-entry-key-attr-values`, and `deletedEntryAttrs` are filtered out through access control instructions.

> **Note**
> This property is only available for notification mode and can be configured using the `create-sync-pipe-config` or the `dsconfig` tool.

The source server must be an Ping Identity PingDirectory Server or Nokia 8661 Directory Server, or an Ping Identity PingDirectoryProxy Server or Nokia 8661 Directory Proxy Server that points to an Ping Identity PingDirectory Server or Nokia 8661 Directory Server.

Considerations for access control filtering

- The directory server will not return the changelog entry if the user is not allowed to see the target entry.
- The directory server strips out any attributes that the user is not allowed to see.
- If no changes are left in the entry, no changelog entry will be returned.
- If only some attributes are stripped out, the changelog entry will be returned.
- Access control filtering on a specific attribute value is not supported. Either all attribute values are returned or none.
- If a sensitive attribute policy is used to filter attributes when a client normally accesses the directory server, this policy will not be taken into consideration during notifications since the Sync User is always connecting using the same method. Configure access controls to filter out attributes, not based on the type of connection made to the server, but based on who is accessing the data. The `filter-changes-by-user` property will be able to evaluate if that person should have access to these attributes.

Configure the Sync Pipe to filter changes by access control instructions

1. Set the `filter-changes-by-user` property to filter changes based on access controls for a specific user.

   ```
   $ bin/dsconfig set-sync-pipe-prop \
     --pipe-name "Notifications Sync Pipe" \
     --set "filter-changes-by-user:uid=admin,dc=example,dc=com"
   ```

2. On the source directory server, set the `report-excluded-changelog-attributes` property to include the names of users that have been removed through access control filtering. This will allow the PingDataSync Server to warn about attributes that were supposed to be synchronized but were filtered out. This step is recommended but not required.

```
$ bin/dsconfig set-backend-prop \
--backend-name "changelog" \
  --set "report-excluded-changelog-attributes:attribute-names"
```

Note

The PingDataSync Server only uses the `attribute-names` setting for the PingDirectory Server’s `report-excluded-changelog-attributes` property. It does not use the `attribute-counts` setting for the property.

Chapter 9: Configure synchronization with SCIM

The PingDataSync Server provides data synchronization between directory servers or proxy servers and System for Cross-domain Identity Management (SCIM) applications over HTTP. Synchronization can be done with custom SCIM applications, or with the PingData PingDirectory Server and PingDirectoryProxy Server configured as SCIM servers using the SCIM extension.

Topics include:

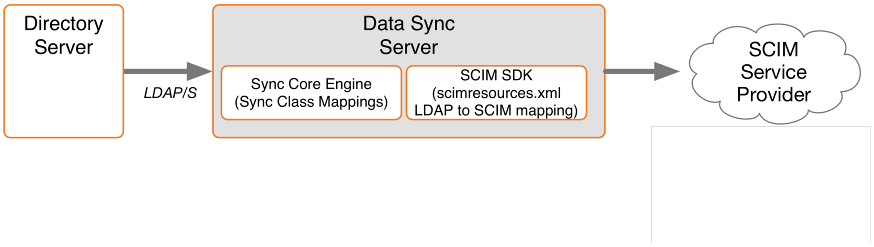
Synchronize with a SCIM Sync Destination overview

The SCIM protocol is designed to make managing user identity in cloud-based applications and services easier. SCIM enables provisioning identities, groups, and passwords to, from, and between clouds. The PingDataSync Server can be configured to synchronize with SCIM service providers.

Note
Both the Ping Identity PingDirectory Server and PingDirectoryProxy Server can be configured to be SCIM servers using the SCIM HTTP Servlet Extension.

The PingDataSync Server is LDAP-centric and operates on LDAP attributes. The SCIM Sync Destination server component acts as a translation layer between a SCIM service provider's schema and an LDAP representation of the entries. While the PingDataSync Server is LDAP-centric and typically at least one endpoint is an LDAP Directory Server, this is not a strict requirement. For example, a JDBC to SCIM sync pipe can be configured.

The PingDataSync Server contains sync classes that define how source and destination entries are correlated. The SCIM Sync Destination contains its own mapping layer, based on `scimresources.xml` that maps LDAP schema to and from SCIM.


Directory Server
LDAP/S
Data Sync Server
Sync Core Engine (Sync Class Mappings)
SCIM SDK (scimresources.xml LDAP to SCIM mapping)
SCIM Service Provider

Synchronizing with a SCIM Sync Destination

Note
The PingDataSync Server can only use SCIM as a Sync Destination. There is no mechanism in the SCIM protocol for detecting changes, so it cannot be used as a Sync Source.

SCIM destination configuration objects

The `SCIMSyncDestination` object defines a SCIM service provider Sync Pipe destination that is accessible over HTTP through the SCIM protocol. It is configured with the following properties:

- `server` – Specifies the names of the SCIM External Servers that are used as the destination of synchronization.

- `resource-mapping-file` – Specifies the path to the `scim-resources.xml` file, a configuration file that defines the SCIM schema and maps it to the LDAP schema. This file is located in `<server root>/config/scim-resources.xml` by default. This file can be customized to define and expose deployment-specific resources.
- `rename-policy` – Specifies how to handle the rename of a SCIM resource.

The SCIM Sync Destination object is based on the SCIM SDK. Before configuring a SCIM destination, review the following documents on the Simple Cloud web site:

- SCIM Core Schema
- SCIM REST API

Considerations for synchronizing to a SCIM destination

When configuring an LDAP to SCIM Sync Pipe, consider the following:

- **Use `scim-resources.xml` for Attribute and DN Mappings** – There are two layers of mapping: once at the Sync Class level and again at the SCIM Sync Destination level in the `scim-resources.xml` file. To reduce complexity, do all possible mappings in the `scim-resources.xml` file.
- **Avoid Groups Unless the SCIM ID is DN Based** – Group synchronization is supported if the SCIM ID is based on the DN. If the SCIM ID is not the DN itself, it must be one of the components of the RDN, meaning that the DNs of group members must contain the necessary attribute.
- **SCIM Modifies Entries Using PUT** – The SCIM Sync Destination modifies entries using the full HTTP PUT method. For every modify, SCIM replaces the entire resource with the updated resource. For information about the implications of this on password updates, see Password Considerations with SCIM.

Renaming a SCIM resource

The SCIM protocol does not support changes that require the SCIM resource to be renamed, such as a MODDN operation. Instead, when a change is detected to an attribute value that is used as part of the SCIM ID attribute, the PingDataSync Server handles it in one of the following ways:

- Deletes the specified SCIM resource and then adds the new resource with the new SCIM ID.
- Adds the new resource with the new SCIM ID and then deletes the old resource.
- Skips the rename portion of the change. If renames are expected on the source endpoint, a careful set of destination-correlation attributes should be chosen so that the destination can still be found after it is renamed on the source.

Configure this by setting the `rename-policy` property of the SCIM Sync Destination.

Password considerations with SCIM

Because the SCIM sync destination modifies entries using a full PUT method, special considerations need to be made for password attributes. An Ping Identity SCIM Server allows password attributes to be omitted from a change when they have not been modified by an operation. This prevents passwords from inadvertently being overwritten during the PUT operation, which does not include the password attribute. Ideally, other SCIM service providers will not wipe a password because a PUT request does not contain it. Check with the SCIM vendor to confirm this behavior before starting a SCIM sync pipe.

Configure synchronization with SCIM

Configure synchronization with SCIM using the `create-sync-pipe-config` utility and the `dsconfig` command. Configuring synchronization between an LDAP server and a SCIM service provider includes the following:

- Configure one external server for every physical endpoint.
- Configure the Sync Source server and designate the external servers that correspond to the source server.
- Configure the Sync Destination server and designate the external servers that correspond to the SCIM sync destination.
- Configure the LDAP to SCIM Sync Pipe.
- Configure the Sync Classes. Each Sync Class represents a type of entry that needs to be synchronized. When specifying a Sync Class for synchronization with a SCIM service provider, avoid including attribute and DN mappings. Instead use the Sync Class to specify the operations to synchronize and which correlation attributes to use.
- Set the evaluation order for the Sync Classes to define the processing precedence for each class.
- Configure the `scim-resources.xml` file. If possible, change the `<resourceIDMapping>` element(s) to use whatever the SCIM Service Provider uses as the SCIM ID.
- Set Up Communication for each External Server. Run `prepare-endpoint-server` once for every LDAP external server that is part of the Sync Source.
- Use `realtime-sync` to start the Sync Pipe.

Configure the external servers

Perform the following to configure an external server for each host in the deployment:

1. Configure an PingDirectory Server as an external server, which will later be configured as a Sync Source. On the PingDataSync Server, run the following `dsconfig` command:

```
$ bin/dsconfig create-external-server \
  --server-name source-ds \
  --type ping-identity-ds \
  --set server-host-name:ds1.example.com \
  --set server-port:636 \
  --set "bind-dn:cn=Directory Manager" \
  --set password:secret \
  --set connection-security:ssl \
  --set key-manager-provider:Null \
  --set trust-manager-provider:JKS
```

2. Configure the SCIM server as an external server, which will later be configured as a Sync Destination. The `scim-service-url` property specifies the complete URL used to access the SCIM service provider. The `user-name` property specifies the account used to connect to the SCIM service provider. By default, the value is `cn=Sync User,cn=Root DNs,cn=config`. Some SCIM service providers may not have the user name in DN format.

```
$ bin/dsconfig create-external-server \
  --server-name scim \
  --type scim \
  --set scim-service-url:https://scim1.example.com:8443 \
  --set "user-name:cn=Sync User,cn=Root DNs,cn=config" \
  --set password:secret \
  --set connection-security:ssl \
  --set hostname-verification-method:strict \
  --set trust-manager-provider:JKS
```

Configure the PingDirectory Server Sync Source

Configure the Sync Source for the synchronization network. More than one external server can be configured to act as the Sync Source for failover purposes. If the source is an PingDirectory Server, also configure the following items:

- Enable the changelog password encryption plug-in on any directory server that will receive password modifications. This plugin intercepts password modifications, encrypts the password, and adds an encrypted attribute to the change log entry.
- Configure the `changelog-deleted-entry-include-attribute` property on the changelog backend, so that the PingDataSync Server can record which attributes were removed during a DELETE operation.

Perform the following steps to configure the Sync Source:

1. Run `dsconfig` to configure the external server as the Sync Source. Based on the previous example where the PingDirectory Server was configured as `source-ds`, run the following command:

```
$ bin/dsconfig create-sync-source --source-name source \
  --type ping-identity \
  --set base-dn:dc=example,dc=com \
  --set server:source-ds \
  --set use-changelog-batch-request:true
```

2. Enable the change log password encryption plug-in on any server that will receive password modifications. The encryption key can be copied from the output, if displayed, or accessed from the `<server-root>/bin/sync-pipe-cfg.txt` file, if the `create-sync-pipe-config` tool was used to create the sync pipe.

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Changelog Password Encryption" \
  --set enabled:true \
  --set changelog-password-encryption-key:<key>
```

3. On the PingDataSync Server, set the decryption key used to decrypt the user password value in the change log entries. The key allows the user password to be synchronized to other servers that do not use the same password storage scheme.

```
$ bin/dsconfig set-global-sync-configuration-prop \
  --set changelog-password-decryption-key:ej5u9e39pq-68
```

4. Configure the `changelog-deleted-entry-include-attribute` property on the changelog backend.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set changelog-deleted-entry-include-attribute:objectClass
```

Configure the SCIM Sync Destination

Configure the SCIM Sync Destination to synchronize data with a SCIM service provider. Run the `dsconfig` command:

```
$ bin/dsconfig create-sync-destination \
  --destination-name scim \
  --type scim \
  --set server:scim
```

Configure the Sync Pipe, Sync Classes, and evaluation order

Configure a Sync Pipe for LDAP to SCIM synchronization, create Sync Classes for the Sync Pipe, and set the evaluation order index for the Sync Classes.

Note
The Synchronization mode must be set to Standard. Notification Mode cannot be used with SCIM.

1. Once the source and destination endpoints are configured, configure the Sync Pipe for LDAP to SCIM synchronization. Run the `dsconfig` command to configure an LDAP-to-SCIM Sync Pipe:

```
$ bin/dsconfig create-sync-pipe \
  --pipe-name ldap-to-scim \
  --set sync-source:source \
  --set sync-destination:scim
```

2. The next set of steps define three Sync Classes. The first Sync Class is used to match user entries in the Sync Source. The second class is used to match group entries. The third class is a DEFAULT class that is used to match all other entries.

   Run the `dsconfig` command to create the first Sync Class and set the Sync Pipe Name and Sync Class name:

   ```
   $ bin/dsconfig create-sync-class \
     --pipe-name ldap-to-scim \
     --class-name user
   ```

3. Use `dsconfig` to set the base DN and filter for this Sync Class. The `include-base-dn` property specifies the base DN in the source, which is `ou=people,dc=example,dc=com` by default. This Sync Class is invoked only for changes at the `ou=people` level. The `include-filter` property specifies an LDAP filter that tells the PingDataSync Server to include `inetOrgPerson` entries as user entries. The `destination-correlation-attributes` specifies LDAP attributes that allow the PingDataSync Server to find the destination resource on the SCIM server. The value of this property will vary. See Identifying a SCIM Resource at the Destination Server for details.

   ```
   $ bin/dsconfig set-sync-class-prop \
     --pipe-name ldap-to-scim \
     --class-name user \
     --add include-base-dn:ou=people,dc=example,dc=com \
     --add "include-filter:(objectClass=inetOrgPerson)" \
     --set destination-correlation-attributes:externalId
   ```

4. Create the second Sync Class, which is used to match group entries.

   ```
   $ bin/dsconfig create-sync-class \
     --pipe-name ldap-to-scim \
     --class-name group
   ```

5. For the second Sync Class, set the base DN and the filters to match the group entries.

   ```
   $ bin/dsconfig set-sync-class-prop \
     --pipe-name ldap-to-scim \
     --class-name group \
     --add include-base-dn:ou=groups,dc=example,dc=com \
     --add "include-filter:(|(objectClass=groupOfEntries)\
       (objectClass=groupOfNames)(objectClass=groupOfUniqueNames)\
       (objectClass=groupOfURLs))"
   ```

6. For the third Sync Class, create a DEFAULT Sync Class that is used to match all other entries. To synchronize changes from only user and group entries, set `synchronize-creates`, `synchronize-modifies`, and `synchronize-delete` to `false`.

   ```
   $ bin/dsconfig create-sync-class \
     --pipe-name ldap-to-scim \
   ```

```
  --class-name DEFAULT \
  --set evaluation-order-index:99999 \
  --set synchronize-creates:false \
  --set synchronize-modifies:false \
  --set synchronize-deletes:false
```

7. After all of the Sync Classes needed by the Sync Pipe are configured, set the evaluation order index for each Sync Class. Classes with a lower number are evaluated first. Run `dsconfig` to set the evaluation order index for the Sync Class. The actual number depends on the deployment.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name ldap-to-scim \
  --class-name user \
  --set evaluation-order-index:100
```

Configure communication with the source server(s)

Configure communication between the PingDataSync Server and the LDAP source servers with the `prepare-endpoint-server` tool. If user accounts do not exist, this tool creates the appropriate user account and its privileges. Also, because the source is an PingDirectory Server, this tool enables the change log.

> **Note**
> The `prepare-endpoint-server` tool can only be used on LDAP directory servers. For the SCIM Server, manually create a sync user entry.

Run the `prepare-endpoint-server` command to setup communication between the PingDataSync Server and the source server(s). The tool will prompt for the bind DN and password to create the user account and enable the change log.

```
$ bin/prepare-endpoint-server \
  --hostname ds1.example.com \
  --port 636 \
  --useSSL \
  --trustAll \
  --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
  --syncServerBindPassword "password" \
  --baseDN "dc=example,dc=com" \
  --isSource
```

Start the Sync Pipe

The `realtime-sync` tool sets a specific starting point for real-time synchronization, so that changes made before the current time are ignored.

1. Run the `realtime-sync` tool to set the startpoint for the Sync Source.

```
$ bin/realtime-sync set-startpoint \
  --end-of-changelog \
  --pipe-name ldap-to-scim
```

2. When ready to start synchronization, run the following command:

   ```
   $ bin/realtime-sync start \
     --pipe-name ldap-to-scim \
     --no-prompt
   ```

Map LDAP schema to SCIM resource schema

The resources configuration file is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for standard SCIM Users and Groups resources, and mappings to standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes. It is installed with the PingDirectory Server. This file can be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

The default configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. In all cases, this must be changed to match the attribute that the destination SCIM service provider is using for its SCIM resource ID. For example, if the destination uses the value of the `uid` attribute, modify the `scim-resources.xml` file to change the `resourceIDMapping` as follows:

```
<resourceIDMapping ldapAttribute="uid" />
```

Ideally, this would be an attribute that exists on the source LDAP entry. If not, the PingDataSync Server can construct it using a Constructed Attribute Mapping. For example, the SCIM service provider used the first and last initials of the user, concatenated with the employee ID (given by the `eid` attribute) as the SCIM resource ID. In this case, an attribute mapping would be constructed as follows:

```
$ dsconfig create-attribute-mapping \
  --map-name MyAttrMap \
  --mapping-name scimID \
  --type constructed \
  --set 'value-pattern:{givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid}'
```

This creates an attribute called `scimID` on the mapped entry when processed by the Sync engine. For example, if the user's name was John Smith, with employee ID 12345, then the `scimID` would be `js12345`. Once this is done,configure the `scim-resources.xml` file as follows:

```
<resourceIDMapping ldapAttribute="scimID" />
```

This will cause it to pull out the constructed `scimID` value from the entry and use that at the SCIM resource ID when making requests to the service provider.

> **Note**
> Constructed attribute mappings support multivalued source attributes for conditional (using the `conditional-value-pattern` property) and non-conditional (using the `value-pattern` property) value patterns. Only one of the source attributes that contribute to a given value pattern can be multivalued.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the PingDirectory Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. If using the SCIM HTTP Servlet Extension for the PingDirectory Server, this can be implemented in many ways, such as:

- Create multiple SCIM servlets, each with a unique `resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). Make sure that any attributes that contain binary data are declared using `dataType=binary` in the `scim-resources.xml` file. When using the Identity Access API, make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes that contain binary data. This instructs the server to base64-encode the data before returning it to clients.

If attributes that are not declared as binary in the schema and contain binary data (or just data that is invalid in XML), the server will check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values that include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scimresources.xml` file.

The <resource> element

A `resource` element has the following XML attributes:

- **schema**: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- **name**: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- **mapping**: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A `resource` element contains the following XML elements in sequence:

- **description**: a required element describing the resource.
- **endpoint**: a required element specifying the endpoint to access the resource using the SCIM REST API.
- **LDAPSearchRef**: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- **LDAPAdd**: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- **attribute**: one or more elements specifying the SCIM attributes for the resource.

The <attribute> element

An `attribute` element has the following XML attributes:

- **schema**: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- **name**: a required attribute specifying the name of the SCIM attribute.
- **readOnly**: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is `false`.
- **required**: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is `false`.

An `attribute` element contains the following XML elements in sequence:

- **description**: a required element describing the attribute. Then just one of the following elements:
- **simple**: specifies a simple, singular SCIM attribute.
- **complex**: specifies a complex, singular SCIM attribute.
- **simpleMultiValued**: specifies a simple, multi-valued SCIM attribute.
- **complexMultiValued**: specifies a complex, multi-valued SCIM attribute.

The <simple> element

A `simple` element has the following XML attributes:

- **dataType**: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `decimal`, `integer`, and `string`.
- **caseExact**: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is `false`.

A `simple` element contains the following XML elements in sequence:

- **mapping**: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

The <complex> element

The `complex` element does not have any XML attributes. It contains the following XML element:

- **subAttribute**: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard type, primary, and display sub-attributes do not need to be specified.

The <simpleMultiValued> element

A `simpleMultiValued` element has the following XML attributes:

- **childName**: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- **dataType**: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, and `string`.
- **caseExact**: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is `false`.

A `simpleMultiValued` element contains the following XML elements in sequence:

- **canonicalValue**: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- **mapping**: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

The <complexMultiValued> element

A `complexMultiValued` element has the following XML attributes:

- **tag**: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard `addresses` SCIM attribute is `address`.

A `complexMultiValued` element contains the following XML elements in sequence:

- **subAttribute**: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.
- **canonicalValue**: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

The <subAttribute> element

A `subAttribute` element has the following XML attributes:

- **name**: a required element specifying the name of the sub-attribute.
- **readOnly**: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is `false`.
- **required**: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is `false`.
- **dataType**: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, and `string`.
- **caseExact**: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is `false`.

A `subAttribute` element contains the following XML elements in sequence:

- **description**: a required element describing the sub-attribute.
- **mapping**: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the `complexMultiValued` element.

The <canonicalValue> element

A `canonicalValue` element has the following XML attributes:

- **name**: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML elements in sequence:

- **subMapping**: an optional element specifying mappings for one or more of the subattributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

The <mapping> element

A `mapping` element has the following XML attributes:

- **ldapAttribute**: a required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- **transform**: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP, and LDAP to SCIM. The available transformations are described in Mapping LDAP Schema to SCIM Resource Schema.

The <subMapping> element

A `subMapping` element has the following XML attributes:

- **name**: a required element specifying the name of the sub-attribute that is mapped.
- **ldapAttribute**: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- **transform**: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. Available transformations are described in Mapping LDAP Schema to SCIM Resource Schema.

The <LDAPSearch> element

A `LDAPSearch` element has the following XML attributes:

- **baseDN**: a required element specifying the LDAP search base DN to be used when querying for the SCIM resource.
- **filter**: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- **resourceIDMapping**: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN.

Note

The `LDAPSearch` element can be added as a top-level element outside of any `<Resource>` elements, and then referenced within them with an `ID` attribute.

The <resourceIDMapping> element

A `resourceIDMapping` element has the following XML attributes:

- **ldapAttribute**: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- **createdBy**: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `<scim-consumer>`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or directory, (as would be the case if the mapped LDAP attribute is `entryUUID`).

If the LDAP attribute value is not listed as destination correlation attribute, this setting is not used by the PingDataSync Server.

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

The <LDAPAdd> element

A `LDAPAdd` element has the following XML attributes:

- **DNTemplate**: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- **fixedAttribute**: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

The <fixedAttribute> element

A `fixedAttribute` element has the following XML attributes:

- **ldapAttribute**: a required attribute specifying the name of the LDAP attribute for the fixed values.
- **onConflict**: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The default value `merge` indicates that the fixed values should be merged with the existing values. The value `overwrite` indicates that the existing values are to be overwritten by the fixed values. The value `preserve` indicates that no changes should be made.

A fixedAttribute element contains the following XML element:

- **fixedValue**: one or more elements specifying the fixed LDAP values.

Identify a SCIM resource at the destination

When a SCIM Sync Destination needs to synchronize a change to a SCIM resource on the destination SCIM server, it must first fetch the destination resource. If the destination resource ID is known, the resource will be retrieved by its ID. If not, a search is performed using the mapped destination correlation attributes. Configuring this requires coordination between the Sync Class and the scim-resources.xml mapping file.

The scim-resources.xml mapping file treats the value of the <resourceIDMapping> element's ldapAttribute attribute as the SCIM ID of the source entry. If this value is also listed as a value of the Sync Class's destination-correlation-attributes property, then the value of this LDAP attribute is used as the SCIM ID of the destination resource.

If no value of destination-correlation-attributes matches the <resourceIDMapping> element's ldapAttribute attribute, the SCIM ID of the destination resource is considered unknown. In this case, the SCIM Sync Destination treats the values of destination-correlation-attributes as search terms, using them to construct a filter for finding the destination resource. Each value of destination-correlation-attributes will be mapped to a corresponding SCIM attribute name, and equality matches will be used in the resulting filter.

If the ldapAttribute value is not listed as a destination correlation attribute, this setting is not used by the PingDataSync Server.

The following table illustrates an LDAPSearch element that contains a resourceIDMapping element:

Identifying a SCIM Resource

Method for Retrieving SCIM Resource	Condition	Example Condition	Example Request
Retrieve resource directly	Used if a destination-correlation-attribute value matches the <resourceIDMapping> ldapAttribute value.	destination-correlation-attribute=mail,uid;<resourceIDMapping ldapAttribute="mail" createdBy= "directory"/>	GET scim/Users/ person@example.com
Retrieve resource using search	Used if no destination-correlation-attribute value matches the <resourceIDMappin	destination-correlation-attribute=mail,uid;<resourceIDMapping ldapAttribute="entryUUID"createdBy= "directory"/>	GET /scim/Users? filter=emails+eq +"person@example.com" and+userName +eq"person"

Identifying a SCIM Resource

Method for Retrieving SCIM Resource	Condition	Example Condition	Example Request
	g> ldapAttribute value.		

The unique ID of a destination SCIM resource will most likely be unknown, and the search method will need to be used. However, not all SCIM service providers support the use of filters. Therefore, not all SCIM service providers may be usable as SCIM Sync Destinations.

Chapter 10: Manage logging, alerts, and alarms

Each PingData server supports extensive logging features to track all aspects of the PingData topology.

Topics include:

Logs and Log Publishers

PingData servers support different types of log publishers that can be used to provide the monitoring information for operations, access, debug, and error messages that occur during normal server processing. The server provides a standard set of default log files as well as mechanisms to configure custom log publishers with their own log rotation and retention policies.

Types of Log Publishers

Several types of log publishers can be used to log processing information about the server, including:

- **Audit loggers** – provide information about actions that occur within the server. Specifically, this type of log records all changes applied, detected or failed; dropped operations that were not completed; changes dropped due to being out of scope, or no changes needed for an operation. The log also shows the entries that were involved in a process.
- **Error loggers** – provide information about warnings, errors, or significant events that occur within the server.
- **Debug loggers** – provide detailed information about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database.
- **Access loggers** – provide information about LDAP operations processed within the server. This log only applies to operations performed in the server. This includes configuration changes, searches of monitor data, and bind operations for authenticating administrators using the command-line tools and the Administrative Console.

View the list of log publishers

View the list of log publishers on each server using the `dsconfig` tool:

```
$ bin/dsconfig list-log-publishers

Log Publisher                      : Type              : enabled
-----------------------------------:-------------------:--------
Debug ACI Logger                   : debug-access      : false
Expensive Operations Access Logger : file-based-access : false
Failed Operations Access Logger    : file-based-access : true
File-Based Access Logger           : file-based-access : true
File-Based Audit Logger            : file-based-audit  : false
File-Based Debug Logger            : file-based-debug  : false
File-Based Error Logger            : file-based-error  : true
Replication Repair Logger          : file-based-error  : true
```

Log compression

PingData servers support the ability to compress log files as they are written. Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled when the logger is created. Compression cannot be turned on or off once the logger is configured. If the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append it to the previous file.

Compression is performed using the standard gzip algorithm. Because it can be useful to have an amount of uncompressed log data for troubleshooting, having a second logger defined that does not use compression may be desired.

Configure compression by setting the `compression-mechanism` property to have the value of `gzip` when creating a new logger. See Creating a New Log Publisher for details.

Configure log file encryption

The server supports the ability to encrypt log files as they are written. The encrypt-log configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an .encrypted extension (and if both encryption and compression are enabled, the extension will be `.gz.encrypted`). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The `encrypt-file` tool (or custom code, using the LDAP SDK's `com.unboundid.util.PassphraseEncryptedInputStream`) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the `encryption-settings create` command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the `encryption-settings-definition-id` property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the `rotate-log` tool to force the file to be rotated before attempting to examine it.

The following commands can be used to set log file encryption:

1. Use `dsconfig` to enable encryption for a Log Publisher. In this example, the File-basedAccess Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted
Access" \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-definition-
id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
   --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
   --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...DoneWriting decrypted
data to file '/ds/PingDirectory/decrypted-access' using akey generated
from encryption settings definition
'332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'Success
fully wrote 123,456,789 bytes of decrypted data
```

Synchronization logs and messages

The PingDataSync Server provides a standard set of default log files to monitor the server activity. View this set of logs in the `<server-root>/logs` directory. The following default log files are available.

PingDataSync Server Logs

Log File	Description
access	File-based Access Log that records LDAP operations processed by the PingDataSync Server. Access log records can be used to provide information about problems during operation processing and provide information about the time required to process each operation.
config-audit.log	Records information about changes made to the server configuration in a format that can be replayed using the `dsconfig` tool.
errors	File-based Error Log that provides information about warnings, errors, and significant events that are not errors but occur during server processing.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to

Log File	Description
	`server.out`.
server.pid	Stores the server's process ID.
server.status	Stores the timestamp, a status code, and an optional message providing additional information on the server status.
setup.log	Records messages that occur during the initial server configuration with the `setup` command.
sync	File-based Sync Log that records synchronization operations processed by the server. Specifically, the log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization.
sync-pipe-cfg.txt	Records the configuration changes used with the `bin/create-sync-pipe-config` tool. The file is placed wherever the tool is run. Typically, this is in `<server-root>` or in the `bin` directory.
tools	Holds logs for long running utilities. Current and previous copies of the log are present in the directory.
update.log	Records messages that occur during a server upgrade.

Sync log message types

The PingDataSync Server logs certain types of log messages with the sync log. Message types can be included or excluded from the logger, or added to a custom log publisher.

Sync Log Message Types

Message Type	Description
change-applied	Default summary message. Logged each time a change is applied successfully.
change-detected	Default summary message. Logged each time a change is detected.
change-failed-detailed	Default detail message. Logged when a change cannot be applied. It includes the reason for the failure and details about the change that can be used to manually repair the failure.
dropped-op-type-not-synchronized	Default summary message. Logged when a change is dropped because the operation type (for example, ADD) is not synchronized for the matching Sync Class.
dropped-out-of-scope	Default summary message. Logged when a change is dropped because it does not match any Sync Class.
no-change-needed	Default summary message. Logged each time a change is dropped because the modified source entry is already synchronized with the destination entry.
change-detected-detailed	Optional detail message. Logged each time a change is detected. It includes attribute values for added and modified entries. This information is useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
entry-mapping-details	Optional detail message. Logged each time a source entry (attributes and DN) are mapped to a destination entry. This information is useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
change-applied-detailed	Optional detail message. Logged each time a change is applied. It includes attribute values for added and modified entries. This information is useful for diagnosing

Message Type	Description
	problems, but it causes log files to grow faster, which impacts performance.
change-failed	Optional summary message. Logged when a change cannot be applied. It includes the reason for the failure but not enough information to manually repair the failure.
intermediate-failure	Optional summary message. Logged each time an attempt to apply a change fails. Note that a subsequent retry of applying the change might succeed.

Create a new log publisher

PingData servers provide customization options to create log publishers with the `dsconfig` command.

After creating a new log publisher, configure the log retention and rotation policies. For more information, see Configure log rotation and Configure log retention.

1. Use the `dsconfig` command to create and configure the new log publisher. (If using `dsconfig` in interactive mode, log publishers are created and manged under the Log Publisher menu.) The following example shows how to create a logger that only logs disconnect operations.

   ```
   $ bin/dsconfig create-log-publisher \
     --type file-based-access --publisher-name "Disconnect Logger" \
     --set enabled:true \
     --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
     --set "rotation-policy:Size Limit Rotation Policy" \
     --set "retention-policy:File Count Retention Policy" \
     --set log-connects:false \
     --set log-requests:false --set log-results:false \
     --set log-file:logs/disconnect.log
   ```

   To configure compression on the logger, add the following option to the previous command:

   ```
   --set compression-mechanism: gzip
   ```

   Compression cannot be disabled or turned off once configured for the logger. Determine logging requirements before configuring this option.

2. View log publishers with the following command:

   ```
   $ bin/dsconfig list-log-publishers
   ```

Configuring log signing

PingData servers support the ability to cryptographically sign a log to ensure that it has not been modified. For example, financial institutions require tamper-proof audit logs files to ensure that transactions can be properly validated and ensure that they have not been modified by a third-party entity or internally by an unauthorized person.

When enabling signing for a logger that already exists, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the `validate-file-signature` tool provided in the `bin` directory of the server (or the `bat` directory on Windows systems). Once this property is enabled, disable and then re-enable the log publisher for the changes to take effect. Perform the following steps to configure log signing:

1. Use `dsconfig` to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:false
```

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true
```

3. To validate a signed file, use the `validate-file-signature` tool to check if a signed file has been altered.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

If any validations errors occur, a message displays that is similar to this:

```
One or more signature validation errors were encountered while validating
the contents of file 'logs/audit':
* The end of the input stream was encountered without encountering the end
of an active signature block. The contents of this signed block cannot be
trusted because the signature cannot be verified
```

Configure log retention and log rotation policies

PingData servers enable configuring log rotation and log retention policies.

Log Retention – When any retention limit is reached, the server removes the oldest archived log prior to creating a new log. Log retention is only effective if a log rotation policy is in place. A new log publisher must have at least one log retention policy configured. The following policies are available:

- **File Count Retention Policy** – Sets the number of log files for the server to retain. The default file count is 10 logs. If the file count is set to 1, the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy** – Sets the minimum amount of free disk space. The default free disk space is 500 MB.
- **Size Limit Retention Policy** – Sets the maximum size of the combined archived logs. The default size limit is 500 MB.
- **Custom Retention Policy** – Create a new retention policy that meets the server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy** – Used in a rare event that does not require log deletion.

Log Rotation – When a rotation limit is reached, the server rotates the current log and starts a new log. A new log publisher must have at least one log rotation policy configured. The following policies are available:

- **Time Limit Rotation Policy** – Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every seven days.
- **Fixed Time Rotation Policy** – Rotates the logs every day at a specified time (based on 24-hour). The default time is 2359.
- **Size Limit Rotation Policy** – Rotates the logs when the file reaches the maximum size. The default size limit is 100 MB.
- **Never Rotate Policy** – Used in a rare event that does not require log rotation.

Configure the log rotation policy

Use `dsconfig` to modify the log rotation policy for the access logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configure the log retention policy

Use `dsconfig` to modify the log retention policy for the access logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configure log listeners

The server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed.

Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true</screen>
```

The path specified by the `copy-to-directory` property must exist, and the filesystem containing that directory must have enough space to hold all of the log files that will be written there. The server will automatically monitor free disk space on the target filesystem and will generate administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the `summarize-access-log` tool on a recently-rotated log file and writes its output to a file in a specified location.

This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the `summarize-access-log` tool, including the `file-based-access` and `operation-timing-access` logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of `.summary`. If the `output-directory` property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically be deleted. Though files are generally small in comparison to the log files themselves, make sure that there is enough space available in the specified storage directory. The server automatically monitors free disk space on the filesystem to which the summary files are written.

System alarms, alerts, and gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server available' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a `Condition` property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a `Probable Cause` property and `Alarm Type` property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

PingData servers are compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when Condition and Resource property are the same. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and Administrative Console.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

Alert handlers

Alert notifications can be sent to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The server provides a number of alert handler implementations configured with the `dsconfig` tool, including:

- **Error Log Alert Handler** – Sends administrative alerts to the configured server error logger(s).
- **JMX Alert Handler** – Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. The server uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SNMP Alert Handler** – Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.

If needed, the Server SDK can be used to implement additional, third-party alert handlers.

Configure alert handlers

Alert handlers can be configured with the `dsconfig` tool. PingData servers support JMX, SMTP, and SNMP. Use the `--help` option for a list of configuration options. The following is a sample command to create and enable an SMTP Alert handler from the command line:

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Test alerts and alarms

After alarms and alert handlers are configured, verify that the server takes the appropriate action when an alarm state changes by manually increasing the severity of a gauge. Alarms and alerts can be verified with the `status` tool.

1. Configure a gauge with `dsconfig` and set the `override-severity` property to critical. The following example uses the CPU Usage (Percent) gauge.

   ```
   $ dsconfig set-gauge-prop \
     --gauge-name "CPU Usage (Percent)" \
     --set override-severity:critical
   ```

2. Run the `status` tool to verify that an alarm was generated with corresponding alerts. The `status` tool provides a summary of the server’s current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

   ```
   $ bin/status
   ```

   ```
                           --- Administrative Alerts ---
   Severity : Time           : Message
   ---------:----------------:-----------------------------------------------
   ```

```
-------
Error    : 11/Aug/2016    : Alarm [CPU Usage (Percent). Gauge CPU Usage
(Percent)
         : 15:41:00 -0500 : for Host System has
         :                : a current value of '18.583333333333332'.
         :                : The severity is currently OVERRIDDEN in the
         :                : Gauge's configuration to 'CRITICAL'.
         :                : The actual severity is: The severity is
         :                : currently 'NORMAL', having assumed this
severity
         :                : Mon Aug 11 15:41:00 CDT 2016. If CPU use is
high,
         :                : check the server's current workload and make
any
         :                : needed adjustments. Reducing the load on the
system
         :                : will lead to better response times.
         :                : Resource='Host System']
         :                : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
hours
Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
                                  --- Alarms ---
Severity : Severity   : Condition : Resource    : Details
         : Start Time :           :             :
---------:------------:-----------:-------------:-------------------------
--
Critical : 11/Aug/2016: CPU Usage : Host System : Gauge CPU Usage
(Percent) for
         : 15:41:00   : (Percent) : : Host System
         :  -0500     :           :             : has a current value of
         :            :           :             : '18.785714285714285'.
         :            :           :             : The severity is
currently
         :            :           :             : 'CRITICAL', having
assumed
         :            :           :             : this severity Mon Aug 11
         :            :           :             : 15:49:00 CDT 2016. If
CPU use
         :            :           :             : is high, check the
server's
         :            :           :             : current workload and
make any
         :            :           :             : needed adjustments.
Reducing
         :            :           :             : the load on the system
will
         :            :           :             : lead to better response
times
```

```
Shown are alarms of severity [Warning,Minor,Major,Critical
Use the --alarmSeverity option to filter this list
```

Use the status tool

PingData servers provides the `status` tool, which outputs the health of the server. The `status` tool polls the current health of the server and displays summary information about the number of operations processed in the network. The tool provides the following information:

Status Tool Sections

Status Section	Description
Server Status	Displays the server start time, operation status, number of connections (open, max, and total).
Server Details	Displays the server details including host name, administrative users, install path, server version, and Java version.
Connection Handlers	Displays the state of the connection handlers including address, port, protocol and current state.
Admin Alerts	Displays the 15 administrative alerts that were generated over the last 48-hour period. Limit the number of displayed alerts using the `--maxAlerts` option. For example, status `--maxAlerts 0` suppresses all alerts.

Synchronization-specific status

The `status` tool displays the following information for the PingDataSync Server.

PingDataSync Server Status Information

Status Section	Description
Sync Topology	Displays information about the connected Sync topology and any standby PingDataSync Server instances.
Summary for Sync Pipe	Displays the health status for each Sync Pipe configured on the topology. Status for each Sync Pipe includes the following: • Started – Indicates whether the Sync Pipe has started. • Current Ops Per Second – Lists the current throughput rate in operations per second. • Percent Busy – Lists the number of current operations currently divided by the number of worker threads. • Changes Detected – Lists the total number of changes detected. • Ops Completed Total – Lists the total number of changes detected and completed. • Num Ops In Flight – Lists the number of operations that are in flight. • Num Ops In Queue – Lists the number of operations that are on the input queue waiting to be synchronized. • Source Unretrieved Changes – Lists how many outstanding changes are still in the source change log that have not yet been retrieved by the PingDataSync Server. If this

Status Section	Description
	is greater than zero, it indicates a backlog, because the internal queue is too full to include these changes. • Failed Op Attempts – Lists the number of failed operation attempts. • Poll For Source Changes Count – Lists the number of times that the source has been polled for changes.
Operations Completed for the Sync Pipe	Displays the completed operation statistics for the sync pipe, including the following: • Success – Lists the number of changes that completed successfully. • Out Of Scope – Lists the number of changes that were included in the Sync Pipe, but were dropped because they did not match criteria in a Sync Class. • Op Type Not Synced – Lists the number of changes that completed because the operation type is not synchronized. • No Change Needed – Lists the number of changes that completed because no change was needed. • Entry Already Exists – Lists the number of changes that completed unsuccessfully because the entry already existed for a Create operation. • No Match Found – Lists the number of changes that completed unsuccessfully because no match for an operation (such as Modify) was found. • Multiple Matches Found – Lists the number of changes that completed unsuccessfully because multiple matches for a source entry were found at the destination. • Failed During Mapping – Lists the number of changes that completed unsuccessfully because there was a failure during attribute or DN mapping. • Failed At Resource – Lists the number of changes that completed unsuccessfully because they failed at the source. • Unexpected Exception – Lists the number of changes that completed unsuccessfully because there was an unexpected exception during processing. • Total – Lists the number of operations completed.
Sync Pipe Source Stats	Displays the source statistics for the external server, including the following: • Is Connected – Indicates whether the Sync Source is connected or not. • Connected Server – Indicates the hostname and port number of the connected server. • Successful Connect Attempts – Indicates the number of successful connection attempts. • Failed Connect Attempts – Indicates the number of failed connection attempts. • Forced Disconnects – Indicates the number of forced disconnects. • Root DSE Polls – Indicates the number of polling attempts of the root DSE. • Unretrieved Changes – Indicates the number of unretrieved changes.

Status Section	Description
	- Entries Fetched – Indicates the number of entries fetched from the source. - Failed To Decode Changelog Entry – Indicates the operations that failed to decode changelog entries. - Ops Excluded By Modifiers Name – Indicates the number of operations excluded by modifier’s name. - Num Backtrack Batches Retrieved – Indicates the number of backtrack batches retrieved.
Sync Pipe Destination Stats	Displays the destination statistics for the external server, including the following: - Is Connected – Indicates whether the Sync Source is connected or not. - Connected Server – Indicates the connection URL of the connected server. - Successful Connect Attempts – Indicates the number of successful connection attempts. - Failed Connect Attempts – Indicates the number of failed connection attempts. - Forced Disconnects – Indicates the number of forced disconnects. - Entries Fetched – Indicates the number of entries fetched. - Entries Created – Indicates the number of entries created. - Entries Modified – Indicates the number of entries modified. - Entries Deleted – Indicates the number of entries deleted.

Monitor the PingDataSync Server

The PingDataSync Server exposes its monitoring information under the `cn=monitor` entry. Various tools can be used to surface the server’s information including the PingDataMetrics Server, the Administrative Console, JConsole, LDAP command-line tools, or SNMP. The following information is collected for the PingDataSync Server. To configure the PingData PingDataMetrics Server to display PingDataSync Server data, see the *Ping IdentityPingDataMetrics Server Administration Guide*.

PingDataSync Server Monitoring Component

Component	Description
Active Operations	Provides information about the operations currently being processed by the server including the number of operations, information about the operation, and the number of active persistent searches.
Backend	Provides general information about the state of the server backend, including the backend ID, base DN(s), entry counts, entry count for the `cn=admin` data, writability mode, and whether it is a private backend. The following backend monitors are provided: - adminRoot

Component	Description
	• ads-truststore • alerts • backup • config • monitor • schema • tasks • userRoot
Berkeley DB JE Environment	Provides information about the state of the Oracle Berkeley DB Java Edition database used by the PingDataSync Server backend.
Client Connections	Provides information about all client connections to the server.
Disk Space Usage	Provides information about the disk space available to various components of the server. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the `DiskSpaceConsumer` interface.
Connection Handler	Provides information about the available connection handlers on the server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections.
General	Provides general information about the server, including product name and server version.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages, and operations handled.
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time, and number of processing times between `0ms` and `1ms`.
System Information	Provides general information about the system and the JVM on which the server is running, including host name, operating system, JVM architecture, Java home, and Java version.
Version	Provides information about the product version, including build ID and revision number.
Work Queue	Provides information about the state of the PingDataSync Server work queue, which holds requests until they can be processed by a worker thread. The work queue configuration has a `monitor-queue-time` property set to `true` by default. This logs messages for new operations with a `qtime` attribute included in

Component	Description
	the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations be processed using the administrative thread pool, the requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

Chapter 11: Troubleshooting

There are several ways to troubleshoot issues with the PingDataSync Server.

Topics include:

Synchronization Troubleshooting

Management Tools

Troubleshooting Tools

Use the status Tool

Use the collect-support-data Tool

Use the Sync Log

Troubleshoot synchronization failures

Installation and maintenance

Problems with SSL communication

Conditions for automatic server shutdown

Enable JVM debugging

Insufficient memory errors

Synchronization troubleshooting

The majority of synchronization problems involve the connection state of the external servers and the synchronization of the data between the two endpoints. Make sure the PingDataSync Server can properly fail over to another endpoint or server instance if the connection fails on the highest priority external server.

Another factor in troubleshooting synchronization is determining if the DN and attribute mappings were properly configured and if the information is properly being synchronized across the network. Typical scenarios include checking for any entry failures and mapping issues.

> **Note**
> Use the `resync` tool to validate Sync Classes and data mappings from one endpoint to another. The tool provides a `dry-run` option that verifies data operations without actually affecting the data.

The following log files are specific to the PingDataSync Server, and contain details about the synchronization processes:

- **Sync Log** – provides information about the synchronization operations that occur within the server. Specifically, the Sync Log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization. The log also shows the entries that were involved in the synchronization process.
- **Sync Failed Operations Log** – provides a list of synchronization operations that have failed.
- **Resync Log** – provides summaries or details of synchronized entries and any missing entries in the Sync Destination.
- **Resync Error Log** – provides error information for `resync` operations.

Management tools

Each PingData server provides command-line tools to manage, monitor, and diagnose server operations. Each tool provides a description of the subcommands, arguments, and usage examples needed to run the tool.

> **Note**
> For detailed information and examples of the command-line tools, see the Configuration Reference Guide in the `<server-root>/docs` directory, or linked from the Administrative Console.

To view detailed argument options and examples, use `--help` with the each tool:

```
$ bin/dsconfig --help
```

For those utilities that support additional subcommands (such as `dsconfig`), list the subcommands with the following:

```
$ bin/dsconfig --help-subcommands
```

View more detailed subcommand information by using `--help` with the specific subcommand:

```
$ bin/dsconfig list-log-publishers --help
```

Troubleshooting tools

PingData provides utilities to troubleshoot the state of each server and to determine the cause of any issues. The following tools are available for diagnosing any problems and are located in the `<server-root>/bin` directory, or the `<server-root>/bat` directory on Windows systems:

Troubleshooting Tools

Tool	Description
`status`	Provides a high-level view of the current operational state of the server and displays any recent alerts that have occurred in past 24 hours.
`ldap-diff`	Used to compare one or more entries across two server endpoints to determine data issues.
`ldapsearch`	Retrieves the full entries from two different servers to determine the exact content of an entry from each server.
`logs`	The logs directory provides important logs that should be used to troubleshoot or monitor any issue with the server. Logs include server-specific operations and the following general logs: • Error Log – Provides information about warnings, errors, or significant events that occur within the server. • Debug Log – Provides detailed information, if enabled, about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database. • Access loggers – Provide information about LDAP operations processed within the server. This log only applies to operations performed in the server. This includes configuration changes, searches of monitor data, and bind operations for authenticating administrators using the command-line tools and the Administrative Console.
`collect-support-data`	Used to aggregate the results of various support tools data for the Ping IdentitySupport team to diagnose. For more information, see Using the collect-support-data Tool.
`config-diff`	Generate a summary of the configuration changes in a local or remote server instance. The tool can be used to compare configuration settings when troubleshooting issues, or when verifying configuration settings on new servers.

Use the status tool

PingData servers provides the `status` tool, which outputs the health of the server. The `status` tool polls the current health of the server and displays summary information about the number of operations processed in the network. The tool provides the following information:

Status Tool Sections

Status Section	Description
Server Status	Displays the server start time, operation status, number of connections (open, max, and total).
Server Details	Displays the server details including host name, administrative users, install path, server version, and Java version.
Connection Handlers	Displays the state of the connection handlers including address, port, protocol and current state.
Admin Alerts	Displays the 15 administrative alerts that were generated over the last 48-hour period. Limit the number of displayed alerts using the `--maxAlerts` option. For example, status `--maxAlerts 0` suppresses all alerts.

Use the collect-support-data tool

PingData servers provide information about their current state and any problems encountered. If a problem occurs, run the `collect-support-data` tool in the `/bin` directory. The tool aggregates all relevant support files into a zip file that can be sent to a support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools for the operating system.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool may vary between systems. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Perform the following steps to run this tool:

1. Navigate to the server root directory.
2. Run the `collect-support-data` tool. Include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data \
  --hostname 100.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager"
  --bindPassword secret \
  --serverRoot /opt/PingData<server> \
  --pid 1234
```

3. Email the zip file to a support provider.

Use the Sync log

The Sync log, located in the logs directory (`<server-root>/logs/sync`), provides useful troubleshooting information on the type of operation that was processed or completed. Most log entries provide the following common elements in their messages:

Sync Log Elements

Sync Log Element	Description
category	Indicates the type of operation, which will always by SYNC.
severity	Indicates the severity type of the message: INFORMATION, MILD_WARNING, SEVERE_WARNING, MILD_ERROR, SEVERE_ERROR, FATAL_ERROR, DEBUG, or NOTICE.
msgID	Specifies the unique ID number assigned to the message.
op	Specifies the operation number specific to the PingDataSync Server.
changeNumber	Specifies the change number from the source server assigned to the modification.
replicationCSN	Specifies the replication change sequence number from the source server.
replicaID	Specifies the replica ID from the source server if there are multiple backend databases.
pipe	Specifies the Sync Pipe that was used for this operation.
msg	Displays the result of this operation.

Sync log example 1

The following example displays an informational message that a modification to an entry was detected on the source server.

```
$ tail -f logs/sync

[17/May/2015:15:46:19 -0500] category=SYNC severity=INFORMATION
msgID=1893728293
op=14 changeNumber=15 replicationCSN=00000128A7E3C7D31E960000000F
replicaID=7830
pipe="DS1 to DS2" msg="Detected MODIFY of uid=user.993,ou=People,dc=example,
dc=com at ldap://server1.example.com:1389"
```

Sync log example 2

The next example shows a successful synchronization operation that resulted from a MODIFY operation on the source server and synchronized to the destination server.

```
[18/May/2015:13:54:04 -0500] category=SYNC severity=INFORMATION
msgID=1893728306 op=701
changeNumber=514663 replicationCSN=00000128ACC249A31E960007DA67 replicaID=7830
pipe="DS1 to DS2" class="DEFAULT" msg="Synchronized MODIFY of
uid=user.698,ou=People,
dc=example,dc=com at ldap://server1.example.com:1389 by modifying entry
uid=user.698,
ou=People,dc=example,dc=com at ldap://server3.example.com:3389"
```

Sync log example 3

The next example shows a failed synchronization operation on a MODIFY operation from the source server that could not be synchronized on the destination server. The log displays the LDIF-formatted modification that failed, which came from a schema violation that resulted from an incorrect attribute mapping (`telephoneNumber -> telephone`) from the source to destination server.

```
[18/May/2015:11:29:49 -0500] category=SYNC severity=SEVERE_WARNING
msgID=1893859389
op=71831 changeNumber=485590 replicationCSN=00000128AC3DE8D51E96000768D6
replicaID=7830 pipe="DS1 to DS2" class="DEFAULT" msg="Detected MODIFY of
uid=user.941,ou=People,dc=example,dc=com at ldap://server1.example.com:1389,
but
failed to apply this change because: Failed to modify entry uid=user.941,
ou=People,dc=example,dc=com on destination 'server3.example.com:3389'.
Cause: LDAPException(resultCode=65(object class violation), errorMessage='
Entry uid=user.941,ou=People,dc=example,dc=com cannot be modified because the
resulting entry would have violated the server schema: Entry
uid=user.941,ou=People,
dc=example,dc=com violates the Directory Server schema configuration because
it
includes attribute telephone which is not allowed by any of theobjectclasses
defined in that entry') (id=1893859386
ResourceOperationFailedException.java:125
Build revision=6226). Details: Source change detail:

dn: uid=user.941,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: 027167170433915
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
Equivalent destination changes:
dn: uid=user.941,ou=People,dc=example,dc=com
changetype: modify
replace: telephone
telephone: 818002279103216
Full source entry:
dn: uid=user.941,ou=People,dc=example,dc=com
objectClass: person
... (more output)
Mapped destination entry:
dn: uid=user.941,ou=People,dc=example,dc=com
telephone: 818002279103216
objectClass: person
```

```
objectClass: inetOrgPerson
... (more output) ...
```

Troubleshoot synchronization failures

While many PingDataSync Server issues are deployment-related and are directly affected by the hardware, software, and network structure used in the synchronization topology, most failures usually fall into one of the following categories:

- **Entry Already Exists** – When an add operation was attempted on the destination server, an entry with the same DN already exists.
- **No Match Found** – A match was not found at the destination based on the current Sync Classes and correlation rules (DN and attribute mapping). When this value has a high count, correlation rule problems are likely.
- **Failure at Resource** – Indicates that some other error happened during the synchronization process that does not fall into the first two categories. Typically, these errors are communication problems with a source or destination server.

Statistics for these and other types of errors are kept in the `cn=monitor` branch and can be viewed directly using the `status` command.

Troubleshoot "Entry Already Exists" failures

If there is a count for the `Entry Already Exists` statistic using the `status` tool, verify the problem in the sync log. For example, the `status` tool displays the following information:

```
           --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 1
No Match Found         : 1
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 0
Unexpected Exception   : 0
Total                  : 2
```

Verify the change by viewing the `<server-root>/logs/sync` file to see the specific operation, which could be due to someone manually adding the entry on the target server:

```
op=2 changeNumber=529277 replicationCSN=00000128AD0D9BA01E960008137D
replicaID=7830
pipe="DS1 to DS2" class="DEFAULT" msg="Detected ADD of
uid=user.1001,ou=People,
dc=example,dc=com at ldap://server1.example.com:1389, but cannot create this
```

```
entry
at the destination because an equivalent entry already exists at
ldap://server3.
example.com:3389. Details: Search using [search-criteria dn:
uid=user.1001,ou=People,
dc=example,dc=com attrsToGet: [*, dn]] returned results;
[uid=user.1001,ou=People,
dc=example,dc=com]. "
```

Perform the following steps to troubleshoot this type of problem:

1. Assuming that a possible DN mapping is ill-formed, first run the `ldap-diff` utility to compare the entries on the source and destination servers. Then look at the `ldap-diff` results with the mapping rules to determine why the original search did not find a match.

   ```
   $ bin/ldap-diff \
     --outputLDIF config-difference.ldif \
     --baseDN "dc=example,dc=com" \
     --sourceHost server1.example.com \
     --targetHost server2.example.com \
     --sourcePort 1389 \
     --targetPort 3389 \
     --sourceBindDN "cn=Directory Manager" \
     --sourceBindPassword password \
     --searchFilter "(uid=1234)"
   ```

2. Review the destination server access logs to verify the search and filters used to find the entry. Typically, the key correlation attributes are not synchronized.
3. If the mapping rule attributes are not synchronized, review the Sync Classes and mapping rules, and use the information from the `ldap-diff` results to determine why a specific attribute may not be getting updated. Some questions to answer are as follows:
   - Is there more than one Sync Class that the operation could match?
   - If using an `include-base-dn` or `include-filter` in the mapping rules, does this exclude this operation by mistake?
   - If using an attribute map, are the mappings correct? Usually, the cause of this error is in the destination mapping attribute settings. For example, if a set of correlation attributes is defined as: `dn`, `mobile`, `accountNumber`, and the `accountNumber` changes for some reason, future operations on this entry will fail. To resolve this, you either remove `accountNumber` from the rule, or add a second rule as: `dn`, `mobile`. The second rule is used only if the search using the first set of attributes fails. In this case, the entry is found and the `accountNumber` information is updated.
4. If deletes are being synchronized, check to see if there was a previous delete of this entry that was not synchronized properly. In some cases, simpler logic should be used for deletes due to the available attributes in the change logs. This scenario could cause

an entry to not be deleted for some reason, which would cause an issue when a new entry with the same DN is added later. Use this information for mapping rules to see why the original search did not find a match.

5. Look at the destination directory server access logs to verify the search and filters it used to find the entry. Typically, the key attribute mappings are not synchronized.

Troubleshoot "No Match Found" failures

If there is a count for the No Match Found statistic using the `status` tool, verify the problem in the sync log. For example, if the `status` tool displays the following:

```
              --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 1
No Match Found         : 1
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 0
Unexpected Exception   : 0
Total                  : 2
```

Verify the change in the `<server-root>/logs/sync` file to see the specific operation:

```
[12/May/2016:10:30:45 -0500] category=SYNC severity=MILD_WARNING
msgID=1893793952
op=4159648 changeNumber=6648922 replicationCSN=4beadaf4002f21150000
replicaID=8469-
ou=test,dc=example,dc=com pipe="DS1 to DS2" class="Others" msg="Detected
DELETE of
'uid=1234,ou=test,dc=example,dc=com' at ldap://server1.example.com:389, but
cannot
DELETE this entry at the destination because no matching entries were found at
ldap://
server2.example.com:389. Details: Search using [search-criteria dn:
uid=1234,ou=test,dc=alu,dc=com filter: (nsUniqueId=3a324c60-5ddb11df-80ffe681-
717b93af) attrsToGet: [*, accountNumber, dn, entryuuid, mobile, nsUniqueId,
object-
Class]] returned no results."
```

Perform the following steps to fix the issue:

1. Test the search using the filter in the error message, if displayed. For example, if the sync log specifies `filter: (nsUniqueId=3a324c60-5ddb11df-80ffe681-717b93af)`, use the `ldapsearch` tool to test the filter. If it is successful, is there anything in the

attribute mappings that would exclude this from working properly?

2. Test the search using the full DN as the base. For example, use `ldapsearch` with the full DN `(uid=1234,ou=People,dc=example,dc=com)`. If it is successful, does the entry contain the attribute used in the mapping rule?
3. If the attribute is not in the entry, determine if there is a reason why this value was not synchronized. Look at the attribute mappings and the filters used in the Sync Classes.

Troubleshoot "Failed at Resource" failures

If there is a count for the "Failed at Resource" statistic using the `status` tool, verify the problem in the sync log. For example, if the `status` tool displays the following information:

```
             --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 0
No Match Found         : 0
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 1
Unexpected Exception   : 0
Total                  : 1
```

This will register after a change is detected at the source in any of the following cases:

- If the fetch of the full source entry fails. The entry exists but there is a connection failure, server down, timeout, or something similar.
- If the fetch of the destination entry fails or if the modification to the destination fails for an exceptional reason (but not for "Entry Already Exists," "Multiple Matches Found," or "No Match Found" issues).

Verify the change by viewing the `<server-root>/logs/sync` file to see the specific operation. If any of the following result codes are listed , the server is experiencing timeout errors:

- `resultCode=timeout: errorMessage=A client-side timeout was encountered while waiting 60000ms for a search response from server server1.example.com:1389`
- `resultCode=timeout: errorMessage=An I/O error occurred while trying to read the response from the server`
- `resultCode=server down: errorMessage=An I/O error occurred while trying to read the response from the server`

- `resultCode=server down: errorMessage=The connection to server server1.example.com:1389 was closed while waiting for a response to search request SearchRequest`
- `resultCode=object class violation: errorMessage='Entry device=1234,dc=example,dc=com violates the Directory Server schema configuration because it contains undefined object class`

With these "Failure at Destination" timeout errors, look at the following settings in the PingDirectory Server to determine if adjustments are needed:

1. For External Server Properties, check the `connect-timeout` property. This property specifies the maximum length of time to wait for a connection to be established before giving up and considering the server unavailable.
2. For the Sync Destination/Sync Source Properties, check the `response-timeout` property. This property specifies the maximum length of time that an operation should be allowed to be blocked while waiting for a response from the server. A value of zero indicates that there should be no client-side timeout. In this case, the server’s default will be used.

```
$ bin/dsconfig --no-prompt --port 389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password list-external-servers \
  --property connect-timeout
```

```
External Server         : Type              : connect-timeout : response-
                                                                timeout
------------------------:-------------------:-----------------:-----------
server1.example.com:389 : sundsee-ds        : 10 s            : -
server2.example.com:389 : sundsee-ds        : 10 s            : -
server3.example.com:389 : ping-identity-ds  : 10 s            : -
server4.example.com:389 : ping-identity-ds  : 10 s            : -
```

3. For Sync Pipe Properties, check the `max-operation-attempts`, `retry-backoff-initialwait`, `retry-backoff-max-wait`, `retry-backoff-increase-by`, `retry-backoff-percentage-increase`. These Sync Pipe properties provide tuning parameters that are used in conjunction with the timeout settings. When a Sync Pipe experiences an error, it will use these settings to determine how often and quickly it will retry the operation.

```
$ bin/dsconfig --no-prompt list-sync-pipes \
  --property max-operation-attempts --property retry-backoff-initial-wait
\
  --property retry-backoff-max-wait --property retry-backoff-increase-by \
  --property retry-backoff-percentage-increase \
  --port 389 --bindDN "cn=Directory Manager" \
  --bindPassword password
```

Installation and maintenance issues

The following are common installation and maintenance issues and possible solutions.

The setup program will not run

If the `setup` tool does not run properly, some of the most common reasons include the following:

A Java Environment Is Not Available – The server requires that Java be installed on the system prior to running the `setup` tool.

If there are multiple instances of Java on the server, run the `setup` tool with an explicitly-defined value for the `JAVA_HOME` environment variable that specifies the path to the Java installation. For example:

```
$ env JAVA_HOME=/ds/java ./setup
```

Another issue may be that the value specified in the provided `JAVA_HOME` environment variable can be overridden by another environment variable. If that occurs, use the following command to override any other environment variables:

```
$ env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected Arguments Provided to the JVM – If the `setup` tool attempts to launch the java command with an invalid set of arguments, it may prevent the JVM from starting. By default, no special options are provided to the JVM when running `setup`, but this might not be the case if either the `JAVA_ARGS` or `UNBOUNDID_JAVA_ARGS` environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, run the following command:

```
$ unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Started – The `setup` tool is only intended to provide the initial configuration for the server. It will not run if it detects that it has already been run.

A previous installation should be removed before installing a new one. However, if there is nothing of value in the existing installation, the following steps can be used to run the `setup` program:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories in the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The server will not start

If the server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running – Only a single instance of the server can run at any time from the same installation root. Other administrative operations can prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The <server> could not acquire an exclusive lock on file
/ds/PingData<server>/locks/server.lock:
The exclusive lock requested for file
/ds/PingData<server>/locks/ server.lock
was not granted, which indicates that another
process already holds a shared or exclusive lock on
that file. This generally means that another instance
of this server is already running.
```

If the server is not running (and is not in the process of starting up or shutting down), and there are no other tools running that could prevent the server from being started, it is possible that a previously-held lock was not properly released. Try removing all of the files in the locks directory before attempting to start the server.

There Is Not Enough Memory Available – When the server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, the server generates an error message indicating that it could not be started.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed, the server might need to be re-configured to use a smaller amount of memory.
- Another process on the system is consuming memory and there is not enough memory to start the server. Either terminate the other process, or reconfigure the server to use a smaller amount of memory.
- The server just shut down and an attempt was made to immediately restart it. If the server is configured to use a significant amount of memory, it can take a few seconds for all of the memory to be released back to the operating system. Run the `vmstat` command and wait until the amount of free memory stops growing before restarting the server.
- If the system is configured with one or more memory-backed filesystems (such as `/tmp`), determine if any large files are consuming a significant amount of memory. If so, remove them or relocate them to a disk-based filesystem.

An Invalid Java Environment or JVM Option Was Used – If an attempt to start the server fails with 'no valid Java environment could be found,' or 'the Java environment could not be started,' and memory is not the cause, other causes may include the following:

- The Java installation that was previously used to run the server no longer exists. Update the `config/java.properties` file to reference the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation has been updated, and one or more of the options that had worked with the previous Java version no longer work. Re-configure the server to use the previous Java version, and investigate which options should be used with the new installation.
- If an `UNBOUNDID_JAVA_HOME` or `UNBOUNDID_JAVA_BIN` environment variable is set, its value may override the path to the Java installation used to run the server (defined in the `config/java.properties` file). Similarly, if an `UNBOUNDID_JAVA_ARGS` environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, explicitly unset the `UNBOUNDID_JAVA_HOME`, `UNBOUNDID_JAVA_BIN`, and `UNBOUNDID_JAVA_ARGS` environment variables before starting the server.

Any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, such as:

```
$ env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option was Used – There are a small number of arguments that can be provided when running the `bin/start-server` command. If arguments were provided and are not valid, the server displays an error message. Correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration – If a change is made to the server configuration using `dsconfig` or the Administrative Console, the server will validate the change before applying it. However, it is possible that a configuration change can appear to be valid, but does not work as expected when the server is restarted.

In most cases, the server displays (and writes to the error log) a message that explains the problem. If the message does not provide enough information to identify the problem, the `logs/config-audit.log` file provides recent configuration changes, or the `config/archived-configs` directory contains configuration changes not made through a supported configuration interface. The server can be started with the last valid configuration using the `--useLastKnownGoodConfig` option:

```
$ bin/start-server --useLastKnownGoodConfig
```

To determine the set of configuration changes made to the server since the installation, use the `config-diff` tool with the arguments `--sourceLocal --targetLocal --sourceBaseline`. The `dsconfig --offline` command can be used to make configuration changes.

Proper Permissions are Missing – The server should only be started by the user or role used to initially install the server. However, if the server was initially installed as a non-root user and then started by the root account, the server can no longer be started as a non-root user. Any new files that are created are owned by root.

If the user account used to run the server needs to change, change ownership of all files in the installation to that new user. For example, if the server should be run as the "ds" user in the "other" group, run the following command as root:

```
$ chown -R ds:other /ds/PingData<server>
```

The server has shutdown

Check the current server state by using the `bin/server-state` command. If the server was previously running but is no longer active, potential reasons may include:

- Shut down by an administrator – Unless the server was forcefully terminated, then messages are written to the error and server logs stating the reason.
- Shut down when the underlying system crashed or was rebooted – Run the `uptime` command on the underlying system to determine what was recently started or stopped.
- Process terminated by the underlying operating system – If this happens, a message is written to the system error log.
- Shut down in response to a serious problem – This can occur if the server has detected that the amount of usable disk space is critically low, or if errors have been encountered during processing that left the server without worker threads. Messages are written to the error and server logs (if disk space is available).
- JVM has crashed – If this happens, then the JVM should provide a fatal error log (a `hs_err_pid<processID>.log` file), and potentially a core file.

The server will not accept client connections

Check the current server state by using the `bin/server-state` command. If the server does not appear to be accepting connections from clients, reasons can include the following:

- The server is not running.
- The underlying system on which the server is installed is not running.
- The server is running, but is not reachable as a result of a network or firewall configuration problem. If that is the case, connection attempts should time out rather than be rejected.
- If the server is configured to allow secure communication through SSL or StartTLS, a problem with the key manager or trust manager configuration can cause connection rejections. Messages are written to the server access log for each failed connection attempt.
- The server may have reached its maximum number of allowed connections. Messages should be written to the server access log for each rejected connection attempt.

- If the server is configured to restrict access based on the address of the client, messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, it can stop listening for new requests. A message should be written to the server error log with information about the problem. Restarting the server can also solve the issue. Another option is to create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The server is unresponsive

Check the current server state by using the `bin/server-state` command. If the server process is running and appears to be accepting connections but does not respond to requests received on those connections, potential reasons for this include:

- If all worker threads are busy processing other client requests, new requests are forced to wait until a worker thread becomes available. A stack trace can be obtained using the `jstack` command to show the state of the worker threads and the waiting requests.

  If all worker threads are processing the same requests for a long time, the server sends an alert that it might be deadlocked. All threads might be tied up processing unindexed searches.
- If a request handler is busy with a client connection, other requests sent through that request handler are forced to wait until it is able to read data. If there is only one request handler, all connections are impacted. Stack traces obtained using the `jstack` command will show that a request handler thread is continuously blocked.
- If the JVM in which the server is running is not properly configured, it can spend too much time performing garbage collection. The effect on the server is similar to that of a network or firewall configuration problem. A stack trace obtained with the `pstack` utility will show that most threads are idle except the one performing garbage collection. It is also likely that a small number of CPUs is 100% busy while all other CPUs are idle. The server will also issue an alert after detecting a long JVM pause that will include details.
- If the JVM in which the server is running has hung, the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a there is a network or firewall configuration problem, communication attempts with the server will fail. A network sniffer will show that packets sent to the system are not receiving TCP acknowledgment.
- If the host system is hung or lost power with a graceful shutdown, the server will be unresponsive.

If it appears that the problem is with the server software or the JVM, work with a support provider to diagnose the problem and potential solutions.

Problems with the Administrative Console

If a problem occurs when trying to use the Administrate Console, reasons may include one of the following:

- The web application container that hosts the console is not running. If an error occurs while trying to start it, consult the logs for the web application container.
- If a problem occurs while trying to authenticate, make sure that the target server is online. If it is, the access log may provide information about the authentication failure.
- If a problem occurs while interacting with the server instance using the Administrative Console, the access and error logs for that instance may provide additional information.

Problems with SSL communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-ds` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Conditions for automatic server shutdown

All PingData servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The PingDirectory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
  --set unrecoverable-database-error-mode:initiate-server-shutdown
```

Insufficient memory errors

If the server shuts down due to insufficient memory errors, it is possible that the allocated heap size is not enough for the amount of data being returned. Consider increasing the heap size, or reducing the number of request handler threads using the following `dsconfig` command:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTP Connection Handler" \
  --set num-request-handlers:<num-of-threads>
```

Enable JVM debugging

Enable the JVM debugging options to track garbage collection data for the system. These options can impact JVM performance, but provide valuable data to tune the server. While the `jstat` utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to provide additional detail, such as:

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

Perform the following steps to enable these options for the server:

1. On the server, navigate to the `config/java.properties` file.
2. Edit the `config/java.properties` file. Add any additional arguments to the end of the line that begins with `start-<server>.java-args`.
3. Save the file.
4. Run the following command for the new arguments to take effect the next time the server is started:

   ```
   $ bin/dsjavaproperties
   ```

Index

PingDirectory™

Release 7.0

Server Administration Guide

Notice

PingDirectory™ Product Documentation

Trademarks

Ping Identity, the Ping Identity logo, PingFederate, PingAccess, and PingOne are registered trademarks of Ping Identity Corporation ("Ping Identity"). All other trademarks or registered trademarks are the property of their respective owners.

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Contents

Chapter 13: Working with Groups.. 203

Chapter 14: Encrypting Sensitive Data...223

Chapter

1

Overview of the Server

Topics:

- *Server Features*
- *Administration Framework*
- *Server Tools Location*

The PingDirectory™ Server is a high performance, extensible directory and PingData Platform, written completely in Java™. The Directory Server centralizes consumer and user identity management information, subscriber data management, application configurations, and user credentials into a network, enterprise, or virtualized database environment. It provides seamless data management over a distributed system based on a standardized solution that meets the constant performance demands for today's markets. The Directory Server simplifies administration, reduces costs, and secures information in systems that scale for very large numbers of users.

This chapter provides an overview of the Directory Server's features and components.

Server Features

Content-Reference to:

The PingDirectory Server is a powerful, 100% Java, production-proven PingData Platform solution for mission-critical and large-scale applications. The Directory Server provides an extensive feature-rich set of tools that can meet the production needs of your system.

- **Full LDAP Version 3 Implementation**. The Directory Server fully supports the Lightweight Directory Access Protocol version 3 (LDAP v3), which supports the Request For Comments (RFCs) specified in the protocol. The Directory Server provides a feature-rich solution that supports the core LDAPv3 protocol in addition to server-specific controls and extended operations.
- **High Availability**. The Directory Server supports N-way multi-master replication that eliminates single points of failure and ensures high availability for a networked topology. The Directory Server allows data to be stored across multiple machines and disk partitions for fast replication. The Directory Server also supports replication in entry-balancing proxy server deployments.
- **Administration Tools**. The Directory Server provides a full set of command-line tools, an Administrative Console, and a Java-based setup tool to configure, monitor, and manage any part of the server. The Directory Server has a task-based subsystem that provides automated scheduling of basic functions, such as backups, restores, imports, exports, restarts, and shutdowns. The set of utilities also includes a troubleshooting support tool that aggregates system metrics into a zip file, which administrators can send to your authorized support provider for analysis.
- **Self-Service Account Manager Application**. The Self Service Account Manager project, hosted at https://github.com/pingidentity/ssam, is a customizable web application allowing users to perform their own account registration, profile updates, and password changes. The project is for testing and development purposes, and is not a supported Ping application.
- **Delegated Admin Application**. A Javascript-based web application can be installed for business users to manage identities stored in the Directory Server. The application provides delegated administration of identities for help desk or customer service representatives (CSR) initiating a password reset and unlock; an employee in HR updating an address stored within another employee profile; or an application administrator updating identity attributes or group membership to allow application SSO access.
- **Security Mechanisms**. The Directory Server provides extensive security mechanisms to protect data and prevent unauthorized access. Access control list (ACL) instructions are available down to the attribute value level and can be stored within each entry. The Directory Server allows connections over Secure Sockets Layer (SSL) through an encrypted communication tunnel. Clients can also use the StartTLS extended operation over standard, non-encrypted ports. Other security features include a privilege subsystem for fine-grained granting of rights, a password policy subsystem that allows configurable password validators and storage schemes, and SASL authentication mechanisms to secure data integrity, such as PLAIN, ANONYMOUS, EXTERNAL, CRAM-MD5, Digest-MD5, and GSSAPI. The Directory Server also supports various providers and mappers for certificate-based authentication in addition to the ability to encrypt specific entries or sensitive attributes. See the *Ping Identity Security Guide* for details.
- **Monitoring and Notifications**. The Directory Server supports monitoring entries using the Data Metrics Server, JConsole, Simple Network Management Protocol (SNMP), or using the Administrative Console. Administrators can track the response times for LDAP operations using a monitoring histogram as well as record performance statistics down to sub-second granularity. The Directory Server also supports configurable notifications, auditing, and logging subsystems with filtered logging capabilities.
- **Powerful LDAP SDK**. The Directory Server is based on a feature-rich LDAP SDK for Java, designed by Ping Identity. The UnboundID® LDAP SDK is a Java API standard that overcomes the many limitations of the Java Naming and Directory Interface (JNDI) model. For example, JNDI does not address the use of LDAP controls and extended operations. The LDAP SDK for Java provides support for controls and extended operations to leverage the Ping Identity's extensible architecture for their applications.

  For more information on the LDAP SDK for Java, to go http://www.LDAP.com.
- **SCIM Extension**. The Directory Server provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure and fast manner.

- **Server SDK**. Ping Identity also provides the Server SDK, which is a library of Java packages, classes, and build tools to help in-house or third-party developers create client extensions for the PingDirectory™ Server, PingDirectoryProxy™ Server, and PingData® Data Sync Server. The servers were designed with a highly extensible and scalable architecture with multiple plug-in points for your customization needs. The Server SDK provides APIs to alter the behavior of each server's components without affecting its code base.

Administration Framework

Content-Reference to:

The Directory Server provides an administration and configuration framework capable of managing stand-alone servers, server groups, and highly-available deployments that include multiple redundant server instances. Administrators can configure changes locally or remotely on a single server or on all servers in a server group. Each server configuration is stored as a flat file (LDIF) that can be accessed under the `cn=config` branch of the Directory Information Tree (DIT). Administrators can tune the configuration and perform maintenance functions over LDAP using a suite of command-line tools, or an Administrative Console (for configuration and monitoring). The Directory Server also provides plug-ins to extend the functionality of its components.

Server Tools Location

Content-Reference to:

Ping Identity distributes the Directory Server, Administrative Console, and LDAP SDK for Java in zip format. After unzipping the file, you can access the `setup` utility in the server root directory, located at `PingDirectory`. The Directory Server stores a full set of command-line tools for maintaining your system in the `PingDirectory/bin` directory for UNIX® or Linux® machines and the `PingDirectory\bat` directory for Microsoft® Windows® machines.

Prior to installing the directory server, read Chapter 2 *Preparing Your Environment*, which presents important information on setting up your machines. Chapter 3 *Installing the Directory Server* presents procedures to install a server instance using the `setup` utility. This utility can be run in one of the two available installation modes: interactive command-line, and non-interactive command-line. Chapter 4 *Configuring the Directory Server* provides procedures to modify the configuration of a server instance or a group of servers using the command-line tools and the Administrative Console.

Chapter

2

Preparing Your Environment

Topics:

The PingDirectory Server offers a highly portable and scalable architecture that runs on multiple platforms and operating systems. The Directory Server is specifically optimized for those operating systems used in environments that process a very large number of entries.

This chapter presents some procedures to set up your server machines for optimal processing efficiency.

Before You Begin

The Directory Server requires certain software packages for the proper operation of the server. For optimized performance, the PingDirectory Server requires Java for 64-bit architectures. To view the minimum required Java version, access your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

It is also highly recommended that a Network Time Protocol (NTP) system be in place so that multi-server environments are synchronized and timestamps are accurate.

Installing Java

For optimized performance, the PingDirectory Server requires Java for 64-bit architectures. You can view the minimum required Java version on your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

Even if your system already has Java installed, you may want to create a separate Java installation for use by the PingDirectory Server to ensure that updates to the system-wide Java installation do not inadvertently impact the Directory Server. This setup requires that the JDK, rather than the JRE, for the 64-bit version, be downloaded.

To Install Java (Oracle/Sun)

1. Open a browser and navigate to the Oracle download site.
2. Download the latest version Java JDK. Click the JDK Download button corresponding to the latest Java update.
3. On the Java JDK page, click the Accept Licence Agreement button, then download the version based on your operating system.

Preparing the Operating System (Linux)

The PingDirectory Server has been extensively tested on multiple operating systems. We have found that several operating system optimizations lead to improved performance. These optimizations include increasing the file descriptor limit on Linux systems, setting filesystem flushes, editing OS-level environment variables, downloading some useful monitoring tools for Redhat Linux systems, and configuring for Huge Page support.

Configuring the File Descriptor Limits

The PingDirectory Server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. If needed, increase the file descriptor limit on the operating system.

If the operating system relies on `systemd`, refer to the Linux operating system documentation for instructions on setting the file descriptor limit.

To Set the File Descriptor Limit (Linux)

The Directory Server allows for an unlimited number of connections by default but is restricted by the file descriptor limit on the operating system. Many Linux distributions have a default file descriptor limit of 1024 per process, which may be too low for the server if it needs to handle a large number of concurrent connections.

Once the operating system limit is set, the number of file descriptors that the server will use can be configured by either using a `NUM_FILE_DESCRIPTORS` environment variable, or by creating a `config/num-file-descriptors` file with a single line such as, `NUM_FILE_DESCRIPTORS=12345`. If these are not set, the default of 65535 is used. This is strictly optional if wanting to ensure that the server shuts down safely prior to reaching the file descriptor limit.

1. Display the current hard limit of your system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

```
ulimit -aH
```

2. Edit the `/etc/sysctl.conf` file. If there is a line that sets the value of the `fs.file-max` property, make sure its value is set to at least 65535. If there is no line that sets a value for this property, add the following to the end of the file:

```
fs.file-max = 65535
```

3. Edit the `/etc/security/limits.conf` file. If the file has lines that sets the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before “#End of file”). Also note that you should insert a tab, rather than spaces, between the columns.

```
* soft nofile 65535
* hard nofile 65535
```

4. Reboot your system, and then use the `ulimit` command to verify that the file descriptor limit is set to 65535.

```
# ulimit -n
```

Note: For RedHat 7 or later, modify the `20-nproc.conf` file to set both the open files and max user processes limits:

```
/etc/security/limits.d/20-nproc.conf

 Add or edit the following lines if they do not already exist:

 *           soft    nproc      65536
 *           soft    nofile     65536
 *           hard    nproc      65536
 *           hard    nofile     65536
 root        soft    nproc      unlimited
```

File System Tuning

Newer ext4 systems use delayed allocation to improve performance. This delays block allocation until it writes data to disk. Delayed allocation improves performance and reduces fragmentation by using the actual file size to improve block allocation. This feature may cause a risk of data loss in cases where a system loses power before all of the data has been written to disk. This may occur if a program is replacing the contents of a file without forcing a write to the disk with `fsync`. Make sure the default `auto_da_alloc` option is enabled on ext4 filesystems.

Administrators can tune ext3 and ext4 filesystems by setting the filesystem flushes and noatime to improve server performance. The following changes can be made in the `/etc/fstab` file.

To Set the Filesystem Flushes

With the out-of-the-box settings on Linux systems running the ext3 filesystem, the data is only flushed to disk every five seconds. If the Directory Server is running on a Linux system using the `ext3` filesystem, consider editing the mount options for that filesystem to include the following:

```
commit=1
```

This variable changes the flush frequency from five seconds to one second.

You should also set the flush frequency to the `/etc/fstab` file. Doing the change via the mount command alone will not survive across reboots.

To Set noatime on ext3 and ext 4 Systems

If you are using an `ext3` or `ext4` filesystem, it is recommended that you set `noatime`, which turns off any *atime* updates during read accesses to improve performance. You should also set the flush frequency to the `/etc/fstab` file. Doing the change via the mount command alone will not survive across reboots.

- Run the following command on an ext3 system.

```
# mount -t ext3 -o noatime /dev/fs1
```

- Run the following command on an ext34 system.

```
# mount -t ext4 -o noatime /dev/fs1
```

Setting the Maximum User Processes

On some Linux distributions (Redhat Enterprise Linux Server/CentOS 6.0 or later), the default maximum number of user processes is set to 1024, which is considerably lower than the same parameter on older distributions (e.g., RHEL/CentOS 5.x). The default value of 1024 leads to some JVM memory errors when running multiple servers on a machine due to each Linux thread being counted as a user process.

At startup, the Directory Server and its tools automatically attempt to raise the maximum user processes limit to 16,383 if the value reported by `ulimit` is less than that. If, for any reason, the server is unable to automatically set the maximum processes limit to 16,383, an error message will be displayed. It is recommended that the limit be set explicitly in `/etc/security/limit.conf`. For example:

```
* soft nproc 65535
* hard nproc 65535
```

The (*) can be replaced with the name of the user under which the software will run. These settings can also be manually configured by setting the `NUM_USER_PROCESSES` environment variable to 16383 or by setting the same variable in a file named `config/num-user-processes`.

About Editing OS-Level Environment Variables

Certain environment variables can impact the Directory Server in unexpected ways. This is particularly true for environment variables that are used by the underlying operating system to control how it uses non-default libraries.

For this reason, the Directory Server explicitly overrides the values of key environment variables like *PATH*, *LD_LIBRARY_PATH*, and *LD_PRELOAD* to ensure that something set in the environments that are used to start the server does not inadvertently impact its behavior.

If there is a legitimate need to edit any of these environment variables, the values of those variables should be set by manually editing the `set_environment_vars` function of the `lib/_script-util.sh` script. You will need to stop (bin/stop-server) and re-start (bin/start-server) the server for the change to take effect.

Install sysstat and pstack (Red Hat)

For Red Hat® Linux systems, you should install a couple of packages, `sysstat` and `pstack`, that are disabled by default, but are useful for troubleshooting purposes in the event that a problem occurs. The troubleshooting tool `collect-support-data` uses the `iostat`, `mpstat`, and `pstack` utilities to collect monitoring, performance statistics, and stack trace information on the server's processes. For Red Hat systems, make sure that these packages are installed, for example:

```
$ sudo yum install sysstat gdb dstat -y
```

Install dstat (SUSE Linux)

The `dstat` utility is used by the `collect-support-data` tool and can be obtained from the OpenSuSE project website. The following example shows how to install the `dstat` utility on SuSE Enterprise Linux 11 SP2:

1. Login as Root.
2. Add the appropriate repository using the `zypper` tool.
3. Install the `dstat` utility.

```
$ zypper install dstat
```

Disable Filesystem Swapping

It is recommended that any performance tuning services like `tuned` be disabled. If performance tuning is required, `vm.swappiness` can be set by cloning the existing performance profile then adding `vm.swappiness = 0` to the new profile's `tuned.conf` file. This file is located at `/usr/lib/tuned/profile-name/tuned.conf`. The updated profile is then selected by running `tuned-adm profile customized_profile`.

Omit vm.overcommit_memory

Administrators should be aware that an improperly configured value for the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file can cause the `setup` or `start-server` tool to fail.

For Linux systems, the `vm.overcommit_memory` property sets the kernel policy for memory allocations. The default value of 0 indicates that the kernel determines the amount of free memory to grant a `malloc` call from an application. If the property is set to a value other than zero, it could lead the operating system to grab too much memory, depriving memory for the `setup` or `start-server` tool.

We recommend omitting the property in the `/etc/sysctl.conf` file to ensure that enough memory is available for these tools.

Managing System Entropy

Entropy is used to calculate random data that is used by the system in cryptographic operations. Some environments with low entropy may have intermittent performance issues with SSL-based communication. This is more typical on virtual machines, but can occur in physical instances as well. Monitor the `kernel.random.entropy_avail` in `sysctl` value for best results.

If necessary, update `$JAVA_HOME/jre/lib/security/java.security` to use `file:/dev/./urandom` for the `securerandom.source` property.

Running as a Non-Root User (Linux)

To run as a non-root user but still allow connections on a privileged port, two options are available:

- **Use a Load-Balancer or Directory Proxy Server**. In many environments, the server can be run on a non-privileged port but can be hidden by a hardware load-balancer or LDAP Directory Proxy Server.
- **Use `netfilter`**. The `netfilter` mechanism, exposed through the `iptables` command, can be used to automatically redirect any requests from a privileged port to the unprivileged port on which the server is listening.

Enabling the Server to Listen on Privileged Ports (Linux)

Linux systems have a mechanism called capabilities that is used to grant specific commands the ability to do things that are normally only allowed for a root account. It may be convenient to enable the server to listen on privileged ports while running as a non-root user.

The `setcap` command is used to assign capabilities to an application. The `cap_net_bind_service` capability enables a service to bind a socket to privileged ports (port numbers less than 1024). If Java is installed in `/ds/java` (and the Java command to run the server is `/ds/java/bin/java`), the Java binary can be granted the `cap_net_bind_service` capability with the following command:

```
$ sudo setcap cap_net_bind_service=+eip /ds/java/bin/java
```

The java binary needs an additional shared library (`libjli.so`) as part of the Java installation. More strict limitations are imposed on where the operating system will look for shared libraries to load for commands that have capabilities assigned. So it is also necessary to tell the operating system where to look for this library. This can be done by creating the file `/etc/ld.so.conf.d/libjli.conf` with the path to the directory that contains the `libjli.so` file. For example, if the Java installation is in `/ds/java`, the contents of that file should be:

```
/ds/java/lib/amd64/jli
```

Run the following command for the change to take effect:

```
$ sudo ldconfig -v
```

Chapter 3

Installing the Server

Topics:

After you have prepared your hardware and software system based on the instructions in Chapter 2, you can begin the setup process using of the PingDirectory Server's easy-to-use installation modes.

This chapter presents the various installation options and procedures available to the administrator.

Getting the Installation Packages

To begin the installation process, obtain the latest ZIP release bundle from Ping Identity and unpack it in a folder of your choice. The release bundle contains the Directory Server code, tools, and package documentation.

To Unpack the Build Distribution

1. Download the latest zip distribution of the Directory Server software.
2. Unzip the compressed zip archive file in a directory of your choice.

```
$ unzip PingDirectory-<version>.zip
```

You can now set up the Directory Server.

About the RPM Package

PingData supports the PingDirectory Server release bundle in an RPM Package Manager (RPM) package for customers who require it. By default, the RPM unpacks the code at /opt/ping-identity/ds/PingDirectory, after which you can run the `setup` command to install the server at that location.

If the RPM install fails for any reason, you can perform an RPM erase if the RPM database entry was created and manually remove the target RPM install directory (e.g., “/opt/ping-identity/ds/PingDirectory” by default). You can install the package again once the system is ready.

To Install the RPM Package

1. Download the latest RPM distribution of the Directory Server software.
2. Unpack the build using the `rpm` command with the --install option. By default, the build is unpacked to `/opt/ping-identity/ds/PingDirectory`. If you want to place the build at another location, use the `--prefix` option and specify the file path of your choice.

```
$ rpm --install PingDirectory-<version>.rpm
```

3. From /opt/ping-identity/ds/PingDirectory/PingDirectory, run the `setup` command to install the server on the machine.

About the Layout of the Directory Server Folders

Once you have unzipped the Directory Server distribution file, you will see the following folders and command-line utilities, shown in the table below.

Table 1: Layout of the Directory Server Folders

Directories/Files/Tools	Description
License.txt	Licensing agreement for the Directory Server.
README	README file that describes the steps to set up and start the Directory Server.
bak	Stores the physical backup files used with the `backup` command-line tool.
bat	Stores Windows-based command-line tools for the Directory Server.
bin	Stores UNIX/Linux-based command-line tools for the Directory Server.
classes	Stores any external classes for server extensions.
collector	Used by the server to make monitored statistics available to the Data Metrics Server.

Directories/Files/Tools	Description
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
db	Stores the Oracle Berkeley Java Edition database files for the Directory Server.
docs	Provides the product documentation.
import-tmp	Stores temporary imported items.
ldif	Stores any LDIF files that you may have created or imported.
legal-notices	Stores any legal notices for dependent software used with the Directory Server.
lib	Stores any scripts, jar, and library files needed for the server and its extensions.
locks	Stores any lock files in the backends.
logs	Stores log files for the Directory Server.
metrics	Stores the metrics that can be gathered for this server and surfaced in the Data Metrics Server.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
revert-update	The `revert-update` tool for UNIX/Linux systems.
revert-update.bat	The `revert-update` tool for Windows systems.
setup	The `setup` tool for UNIX/Linux systems.
setup.bat	The `setup` tool for Windows systems.
scim-data-tmp	Used to create temporary files containing SCIM request data.
uninstall	The `uninstall` tool for UNIX/Linux systems.
uninstall.bat	The `uninstall` tool for Windows systems.
update	The `update` tool for UNIX/Linux systems.
update.bat	The `update` tool for Windows systems.
Velocity	Stores any customized Velocity templates and other artifacts (CSS, Javascript, images), or Velocity applications hosted by the server.

About the Server Installation Modes

One of the strengths of the PingDirectory Server is the ease with which you can install a server instance using the `setup` tool. The `setup` tool allows you to quickly install and configure a stand-alone Directory Server instance.

To install a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode prompts for information during the installation process. To run the installation in this mode, use the `setup --cli` command.
- **Non-Interactive Command-Line Mode**. Non-interactive command-line mode is designed for setup scripts to automate installations or for command-line usage. To run the installation in this mode, `setup` must be run with the --no-prompt option as well as the other arguments required to define the appropriate initial configuration.

All installation and configuration steps should be performed while logged on to the system as the user or role under which the Directory Server will run.

Before You Begin

After you have unzipped the Directory Server ZIP file, you may want to carry out the following functions depending on your deployment requirements:

- **Custom Schema Elements**. If your deployment uses custom schema elements in a custom schema file (for example, `98-schema.ldif`), you may do one of the following:
  - Copy your custom schema file to the `config/schema` directory before running setup.
  - Copy your custom schema file to the `config/schema` directory after setup and re-start the server. If replication is enabled, the restart will result in the schema replicating to other servers in the replication topology.
  - Use the **Schema Editor** after setup. If replication is enabled, schema definitions added through the **Schema Editor** will replicate to all servers in the replication topology without the need for a server restart.
- **Certificates**. If you are setting up a new machine instance, copy your keystore and truststore files to the `<server-root>/config` directory prior to running setup. The keystore and truststore passwords can be placed, in clear text, in corresponding `keystore.pin` and `truststore.pin` files in `<server-root>/config`.
- **Encryption Passphrase**. Encryption for directory data, backups, LDIF exports, and log files can be enabled during setup by providing or generating an encryption key with a passphrase.
- **Locations**. Location names are used to define a grouping of Ping Identity Server products based on physical proximity. For example, a location is most often associated with a single datacenter location. During the installation, assign a location to each server for optimal inter-server behavior. The location assigned to a server within Global Configuration can be referenced by components within the server as well as processes external to the server to satisfy "local" versus "remote" decisions used in replication, load balancing, and failover.
- **Validate ACIs**. Many directory servers allow for less restrictive application of its access control instructions (ACIs), so that they accept invalid ACIs. For example, if a Sun/Oracle server encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the PingDirectory Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a PingDirectory Server. If you are migrating from a Sun/Oracle deployment to a PingDirectory Server, the PingDirectory Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. For more information, see Validating ACIs Before Migrating Data.

Important:

Each Server Deployment Requires an Execution of Setup - Duplicating a Server-root is not Supported. The installation of the server does not write or require any data outside of the server-root directory. After executing `setup`, copying the server-root to another location or system, in order to *duplicate* the installation, is not a supported method of deployment. The server-root can be moved to another host or disk location if a host or file system change is needed.

Ping License Keys

License keys are required to install all Ping products. Obtain licenses through Salesforce or from https://www.pingidentity.com/en/account/request-license-key.html.

- A license is always required for setting up a new single server instance and can be used site-wide for all servers in an environment. When cloning a server instance with a valid license, no new license is needed.
- A new license must be obtained when updating a server to a new major version, for example from 6.2 to 7.0. Licenses with no expiration date are valid until the server is upgraded to the next major version. A prompt for a new license is displayed during the update process.
- A license may expire on particular date. If a license does expire, obtain a new license and install it using `dsconfig` or the Administrative Console. The server will provide a notification as the expiration date approaches. License details are available using the server's `status` tool.

When installing the server, specify the license key file in one of the following ways:

- Copy the license key file to the server root directory before running setup. The interactive `setup` tool will discover the file and not require input. If the file is not in the server root, the `setup` tool will prompt for its location.
- If the license key is not in the server root directory, specify the `--licenseKeyFile` option for non-interactive setup, and the path to the file.

Setting Up the Directory Server in Interactive Mode

The `setup` tool also provides an interactive text-based command-line interface to set up a Directory Server instance.

To Install the Directory Server in Interactive Mode

1. Unzip the distribution ZIP file, review Before You Begin, and then go to the server root directory. Use the `setup` utility with the --cli option to install the server in interactive mode.

   ```
   $ ./setup
   ```

   If the *JAVA_HOME* environment variable is set to an older version of Java, explicitly specify the path to the Java JDK installation during the setup process. Either set the *JAVA_HOME* environment variable with the Java JDK path or execute the `setup` command in a modified Java environment using the `env` command.

   ```
   $ env JAVA_HOME=/ds/java ./setup
   ```
2. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
3. Enter the fully qualified host name or IP address of the local host, or press **Enter** to accept the default.
4. Enter the root user DN, or press **Enter** to accept the default (cn=Directory Manager).
5. Enter and confirm the root user password.
6. Press **Enter** to enable the Ping Identity services (Configuration, Consent, Delegated Admin, Documentation, and SCIM) and Administrative Console over HTTPS. After setup, individual services and applications can be enabled or disabled by configuring the HTTPS Connection Handler.
7. Enter the port on which the Directory Server will accept connections from HTTPS clients, or press **Enter** to accept the default.
8. Enter the port on which the Directory Server will accept connections from LDAP clients, or press **Enter** to accept the default.
9. The next two options enable using LDAPS or StartTLS. Type `no` to use a standard LDAP connection, or accept the default (`yes`) to enable both. Enabling LDAPS configures the LDAPS Connection Handler to allow SSL over its client connections. Enabling StartTLS configures the LDAP Connection Handler to allow StartTLS.
10. Select the certificate option for this server:

    - Generate a self-signed certificate for testing purposes only.
    - To use an existing certificate using a Java Keystore, enter the keystore path and keystore PIN.
    - To use an existing certificate using use a PKCS#12 keystore, enter the keystore path and the keystore PIN.
    - To use the PKCS#11 token, enter only the keystore PIN.
11. Choose the desired encryption for the directory data, backups, and log files from the choices provided:

    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.

- Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
- Do not encrypt server data.

12. Type the base DN for the data, or accept the default base DN of `dc=example,dc=com`.
13. Choose an option to generate and import sample data. Type the desired number of entries, or press `Enter` to accept the default number (10000). This option is used for quick evaluation of the Directory Server.

    See Initializing Data onto the Server if you want to use other options to initialize the server.
14. Choose the option to tune the amount of memory that will be consumed by the Directory Server and its tools.
15. Press `Enter` to prime or preload the database cache at startup prior to accepting client connections.

    Priming the cache can increase the startup time for the Directory Server but provides optimum performance once startup has completed. This option is best used for strict throughput or response time performance requirements, or if other replicas in a replication topology can accept traffic while this Directory Server instance is starting. Priming the cache also helps determine the recommended JVM option, CMSInitiatingOccupancyFraction, when a Java garbage collection pause occurs. See JVM Garbage Collection Using CMS.
16. Enter a location name for this server.
17. Enter a unique instance name for this server. Once set, the name cannot be changed.
18. Press **Enter** to accept the default (yes) to start the Directory Server after the configuration has completed. To configure additional settings or import data, type `no` to keep the server in shutdown mode.
19. Choose an option to continue server set up.
20. On the **Setup Summary** page, confirm the configuration. Press **Enter** to accept the default (set up with the parameters given), enter the option to repeat the installation process, or enter the option to cancel the setup completely.

Installing the Directory Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. Non-interactive mode is useful when setting up production or QA servers with specific configuration requirements.

The non-interactive command-line mode requires that all mandatory options be present for each command call. If there are missing or incorrect arguments, the `setup` tool fails and aborts the process. You must also use a --no-prompt option to suppress interactive output, except for errors, when running in non-interactive mode. Additionally, you must also use the --acceptLicense option and specify the port using the --ldapPort or --ldapsPort option. If neither option is specified, an error message is displayed. To view the license, run bin/review-license command.

To automatically tune the JVM to use maximum memory, use the --aggressiveJVMTuning and --maxHeapSize {memory} options. To preload the database at startup, use the --primeDB option.

To configure a deployment using a truststore, see Installing the Directory Server in Non-Interactive Mode with a Truststore.

To see a description of the available command-line options for the `setup` tool, use `setup --help`.

To Install the Directory Server in Non-Interactive Mode

The following procedure shows how to install a Directory Server in a production or QA environment with no security enabled.

- Unzip the distribution ZIP file, review “Before You Begin”, and then use `setup` with the --cli and --no-prompt options for non-interactive mode from the `<server-root>` directory. The following command uses the default root user DN (cn=Directory Manager) with the specified --rootUserPassword option. You must include the --acceptLicense option or the setup will generate an error message.

  ```
  $ ./setup --cli --no-prompt --rootUserPassword "password" \
    --baseDN "dc=example,dc=com" --acceptLicense --ldapPort 389
  ```

To Install the Directory Server in Non-Interactive Mode with a Truststore

You can set up the Directory Server using an existing truststore for secure communication. This section assumes that you have an existing keystore and truststore with trusted certificates.

- Unzip the distribution ZIP file, review Before You Begin, and then, from the server root directory, use `setup` with the --cli and --no-prompt options for non-interactive mode. The following example enables security using both SSL and StartTLS. It also specifies a JKS keystore and truststore that define the server certificate and trusted CA. The `userRoot` database contents will remain empty and the base DN entry will not be created.

  ```
  $ ./setup --cli --no-prompt --rootUserPassword "password" \
    --baseDN "dc=example,dc=com" --ldapPort 389 --enableStartTLS \
    --ldapsPort 636 --useJavaKeystore config/keystore.jks \
    --keyStorePasswordFile config/keystore.pin \
    --certNickName server-cert --useJavaTrustStore config/truststore.jks \
    --acceptLicense
  ```

  The password to the private key with the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined with the Administrative Console or the `dsconfig` tool by editing the Trust Manager Provider standard configuration object.

Installing a Lightweight Server

Users who want to demo or test a lightweight version of the Directory Server on a memory-restricted machine can do so by removing all unused or unneeded configuration objects. All configuration entries, whether enabled or not, take up some amount of memory to hold the definition and listeners that will be notified of changes to those objects.

The configuration framework will not allow you to remove objects that are referenced, and in some cases if you have one configuration object referencing another but really do not need it, then you will first need to remove the reference to it. If you try to remove a configuration object that is referenced, both `dsconfig` and the Administrative Console should prevent you from removing it and will tell you what still references it.

Depending on your test configuration, some example configuration changes that can be made are as follows:

- **Reduce the number of worker threads**. Each thread has a stack associated with it, and that consumes memory. If you're running a bare-bones server, then you probably do not have enough load to require a lot of worker threads.

  ```
  $ bin/dsconfig set-work-queue-prop \
    --set num-worker-threads:8 \
    --set num-administrative-session-worker-threads:4 \
    --set max-work-queue-capacity:100
  ```
- **Reduce the percentage of JVM memory used for the JE database cache**. When you have a memory-constrained environment, you want to ensure that as much of the memory that is there is available for use during processing and not tied up caching database contents.

  ```
  $ bin/dsconfig set-backend-prop --backend-name userRoot --set db-cache-
  percent:5
  ```
- **Disable the Dictionary Password Validator**. The Dictionary Password Validator takes a lot of memory to hold its dictionary. Disabling it will free up some memory. You can delete the other password validators if not needed, such as Attribute Value, Character Set, Length-based, Repeated Characters, Similarity-based, or Unique Characters Password Validator.

  ```
  $ bin/dsconfig delete-password-validator --validator-name Dictionary
  ```
- **Remove non-essential schema files**. Although not recommended for production deployments, some candidates that you can remove are the following: `03-rfc2713.ldif`, `03-rfc2714.ldif`, `03-rfc2739.ldif`, `03-rfc2926.ldif`, `03-rfc2985.ldif`, `03-rfc3712.ldif`, `03-uddiv3.ldif`.

There are other items that can be removed, depending on your desired configuration. Contact your authorized support provider for assistance.

Running the Status Tool

The Directory Server provides a `status` tool that outputs the current state of the server as well as other information, such as server version, JE Environment statistics, Operation Processing Times, Work Queue, and Administrative Alerts. The `status` tool is located in the `bin` directory (UNIX, Linux) or the `bat` directory (Windows).

To Run the Status Tool

- Run the `status` command on the command line. The following command displays the current Directory Server status and limits the number of viewable alerts in the last 48 hours. It provides the current state of each connection handler, data sources, JE environment statistics, processing times by operation type and current state of the work queue.

```
$ bin/status --bindDN “uid=admin,dc=example,dc=com” --bindPassword secret

          --- Server Status ---
Server Run Status:    Started 28/Mar/2012:10:47:17.000 -0500
Operational Status:   Available
Open Connections:     13
Max Connections:      13
Total Connections:    50

          --- Server Details ---
Host Name:            server1.example.com
Administrative Users: cn=Directory Manager
Installation Path:    PingDirectory
Server Version:       PingDirectory Server 7.0.0.0
Java Version:         jdk-7u9

          --- Connection Handlers ---
Address:Port : Protocol : State
-------------:----------:---------
0.0.0.0:1389 : LDAP     : Enabled
0.0.0.0:1689 : JMX      : Disabled
0.0.0.0:636  : LDAPS    : Disabled

           --- Data Sources ---
Base DN:             dc=example,dc=com
Backend ID:          userRoot
Entries:             2003
Replication:         Enable
Replication Backlog: 0
Age of Oldest Backlog Change: not available

           --- JE Environment ---
ID                 : Cache Full :  Cache   : On-Disk : Alert
-------------------:------------:----------:----------:------
replicationChanges : 6 %     : 328.8 kb : 30.4 kb : None
userRoot        : 9 %        : 6.2mb    : 146.6mb : None

           --- Operation Processing Time ---
Op Type   : Total Ops : Avg Resp Time (ms)
----------:-----------:------------------
Add       : 0         : 0.0
Bind      : 0         : 0.0
Compare   : 0         : 0.0
Delete    : 0         : 0.0
Modify    : 2788567   : 0.921
Modify    : 0         : 0
DN        : 2267266   : 0.242
Search    : 5055833   : 0.616
```

```
All

            --- Work Queue ---
           : Recent : Average : Maximum
-----------:--------:---------:--------
Queue Size : 4   : 0    : 10
% Busy  : 26  : 5    : 100

            --- Administrative Alerts ---
Severity : Time                          : Message
---------:----------------------------:----------------------------------------------
Info     : 28/Mar/2012 10:47:17 -0500 : The Directory Server has started
 successfully
Info     : 28/Mar/2012 10:47:14 -0500 : The Directory Server is starting
Info     : 28/Mar/2012 10:44:22 -0500 : The Directory Server has started
 successfully
Info     : 28/Mar/2012 10:44:18 -0500 : The Directory Server is starting

Shown are alerts of type [Info,Warning,Error,Fatal] from the past 48 hours
 Use the
--maxAlerts and/or --severity options to filter this list
```

Note: By default, the `status` command displays the alerts generated in the last 48 hours. You can limit the number of alerts by using the --maxAlerts option.

Where To Go From Here

After you have set up your Directory Server instance, you can configure any specific server settings, import your user database, or run initial performance tests to optimize your server’s throughput.

- **Log into the Administrative Console**. Become familiar with configuration options through the Administrative Console interface. The URL is based on the hostname and HTTPS port specified during installation, such as https://hostname.com:443/console.
- **Apply Server Configurations**. Apply your server configuration changes individually or using a `dsconfig` batch file. The batch file defines the Directory Server configuration tool, `dsconfig`, commands necessary to configure your server instance. For more information on using batch files, see Using dsconfig in Batch Mode.

  If you are migrating from a Sun Java System 5.x, 6.x, 7.x directory server, you can use the `bin/migrate-sun-ds-config` command to migrate your configuration settings to this newly installed server instance.
- **Import Data**. Import user data using the `import-ldif` tool. The import serves as an initial test of the schema settings.

  ```
  $ bin/import-ldif --backendID userRoot --ldifFile ../user-data.ldif
  ```
- **Install and Configure the Delegated Admin Application**. A Javascript-based web application can be installed for business users to manage identities stored in the Directory Server. The application provides delegated administration of identities for help desk or customer service representatives (CSR) initiating a password reset and unlock; an employee in HR updating an address stored within another employee profile; or an application administrator updating identity attributes or group membership to allow application SSO access.
- **Run Performance Tests**. The Directory Server provides two tools for functional performance testing using in-house LDAP clients that accesses the server directly: `searchrate` (tests search performance) and `modrate` (tests modification performance):

  ```
  $ bin/searchrate --baseDN "dc=example,dc=com" --scope sub \
    --filter "(uid=user.[0-1999])" --attribute givenName --attribute sn \
    --attribute mail --numThreads 10

  $ bin/modrate --entryDN "uid=user.[0-1999],ou=People,dc=example,dc=com" \
    --attribute description --valueLength 12 --numThreads 10
  ```

Working with Multiple Backends

You can create multiple local database backends, each containing one or more different base DNs. There should be at most one replicating domain on each local database backend. The replication domain should not span multiple local database backends. The typical entry-balancing configuration involves two local database backends: one backend to serve the global domain data that resides above the entry-balancing point and a backend that is defined with the entry-balancing point as the base DN, such as `ou=people,dc=example,dc=com`.

With multiple local database backends configured, the data existing with each backend can be managed independently. In addition, separate index settings are applied to each local database backend.

When creating multiple databse backends, consider the following:

- No two backends may have the same base DN.
- If any base DN for a given backend is subordinate to a base DN on another backend, then all base DNs on that backend must be subordinate to the base DN of the other backend.
- The total of all db-cache-percent values should be no more than 65-70% in most cases and should never be configured to exceed 100%.

Importing Data

After installation, the database, such as userRoot, will need to have data imported. For a server to be added to a replicating set, the database will be imported as part of the `dsreplication initialize` operation, which is performed after `dsreplication enable`. A server that will not be added to a replicating set, or the first server of a future replicating set, should have data imported with the `bin/import-ldif` tool. See Chapter 8, Importing and Exporting Data, for more infomation about the `bin/import-ldif` tool.

Generating Sample Data

The PingDirectory Server provides LDIF templates that can be used to generate sample entries to initialize your server. You can generate the sample data with the `make-ldif` utility together with template files that come bundled with the ZIP build, or you can use templates files that you create yourself. The templates create sequential entries that are convenient for testing the PingDirectory Server with a range of dataset sizes. The Directory Server templates are located in the `config/MakeLDIF`.

To randomize the data, the `make-ldif` command has a --randomSeed option that can be used to seed the random number generator. If this option is used with the same seed value, the template will always generate exactly the same LDIF file.

The sample data templates generate a dataset with basic access control privileges that grants anonymous read access to anyone, grants users the ability to modify their own accounts, and grants the Admin account full privileges. The templates also include the uid=admin and ou=People entries necessary for a complete dataset. You can bypass the `make-ldif` command entirely and use the --templateFile option with the `import-ldif` tool.

- Use the `make-ldif` command to generate sample data. The command generates 10,000 sample entries and writes them to an output file, `data.ldif`. The random seed generator is set to 0.

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-10k.template \
  --ldifFile /path/to/data.ldif --randomSeed 0
```

To Import Data on the Directory Server Using Offline Import

1. Create an LDIF file that contains entries, or locate an existing file.

   The import-ldif tool requires an LDIF file, which conforms to standard LDIF syntax without change records. This means the changeType attribute is not allowed in the input LDIF. For information on adding entries to the Directory Server, see Managing Entries.
2. Stop the Directory Server.

3. Use the offline `import-ldif` to import data from an LDIF file to the Directory Server. For assistance with the list of options, run `import-ldif --help`.

   In the following example, the data is imported from the `data.ldif` file to the `userRoot` backend. If any entry is rejected due to a schema violation, then the entry and the reason for the rejection is written to the `rejects.ldif` file. Skipped entries, written to `skipped.ldif`, occur if an entry cannot be placed under a branch node in the DIT or if exclusion filtering is used (`--excludeBranch`, `--excludeAttribute`, or `--excludeFilter`). The `--overwrite` option instructs `import-ldif` to overwrite existing skipped and rejected files. The `--overwriteExistingEntries` option indicates that any existing data in the backend should be overwritten. Finally, the `--stripTrailingSpaces` option strips trailing spaces on attributes that would otherwise result in a LDIF parsing error.

   ```
   $ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif --
   rejectFile
       rejects.ldif --skipFile skipped.ldif --overwrite --
   overwriteExistingEntries --stripTrailingSpaces
   ```
4. Re-start the Directory Server.

Running the Server

To start the Directory Server, run the bin/start-server command on UNIX or Linux systems (an analogous command is in the `bat` folder on Microsoft Windows systems). The bin/start-server command starts the Directory Server as a background process when no options are specified. To run the Directory Server as a foreground process, use the bin/start-server command with the --nodetach option.

To Start the Directory Server

Use bin/start-server to start the server.

```
$ bin/start-server
```

To Run the Server as a Foreground Process

1. Enter bin/start-server with the --nodetach option to launch the Directory Server as a foreground process.

   ```
   $ bin/start-server --nodetach
   ```
2. You can stop the Directory Server by pressing `CNTRL+C` in the terminal window where the server is running or by running the bin/stop-server command from another window.

To Start the Server at Boot Time

By default, the PingDirectory Server does not start automatically when the system is booted. Instead, you must manually start it with the bin/start-server command. To configure the Directory Server to start automatically when the system boots, use the `create-systemd-script` utility to create a script, or create the script manually.

1. Create the service unit configuration file in a temporary location where "ds" is the user the PingDirectory will run as.

   ```
   $ bin/create-systemd-script \
        --outputFile /tmp/ping-directory.service \
        --userName ds
   ```
2. As a root user, copy the `ping-directory.service` configuration file into the `/etc/systemd/system` directory.
3. Reload `systemd` to read the new configuration file.

   ```
   $ systemctl daemon-reload
   ```
4. To start the PingDirectory, use the `start` command.

```
$ systemctl start ping-directory.service
```

5. To configure the PingDirectory to start automatically when the system boots, use the `enable` command.

```
$ systemctl enable ping-directory.service
```

6. Log out as root.

   If on an RC system, this task is done by creating the startup script with `bin/create-rc-script` and moving it to the `/etc/init.d` directory. Create symlinks to it from the `/etc/rc3.d` directory (staring with an “S” to ensure that the server is started) and `/etc/rc0.d` directory (starting with a “K” to ensure that the server is stopped).

Logging into the Administrative Console

After the server is installed, access the Administrative Console, `https://hostname:HTTPport/console/login`, to verify the configuration and manage the server. To log into the Administrative Console, use the initial root user DN specified during setup (by default `cn=Directory Manager`).

The `dsconfig` command or the Administrative Console can be used to create additional root DN users in `cn=Root DNs,cn=config`. These new users require the fully qualified DN as the login name, such as `cn=new-admin,cn=Root DNs,cn=config`. To use a simple user name (with out the `cn=` prefix) for logging into the Administrative Console, the root DN user must have the `alternate-bind-dn` attribute configured with an alternate name, such as "admin."

By default the link to the Administrative Console is `https://hostname:HTTPport/console/login`.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package (`/server-root/resource/admin-console.zip`) can be installed according to that container's documentation.

Stopping the Directory Server

The Directory Server provides a simple shutdown script, bin/stop-server, to stop the server. You can run it manually from the command line or within a script.

If the Directory Server has been configured to use a large amount of memory, then it can take several seconds for the operating system to fully release the memory and make it available again. If you try to start the server too quickly after shutting it down, then the server can fail because the system does not yet have enough free memory. On UNIX systems, run the `vmstat` command and watch the values in the "free" column increase until all memory held by the Directory Server is released back to the system.

You can also set a configuration option that specifies the maximum shutdown time a process may take.

To Stop the Server

- Use the bin/stop-server tool to shut down the server.

```
$ bin/stop-server
```

To Schedule a Server Shutdown

- Use the `bin/stop-server` tool with the --stopTime YYYYMMDDhhmmss option to schedule a server shutdown.

  The Directory Server schedules the shutdown and sends a notification to the `server.out` log file. The following example sets up a shutdown task that is scheduled to be processed on June 6, 2012 at 8:45 A.M. CDT. The server uses the UTC time format if the provided timestamp includes a trailing “Z”, for example, 20120606134500Z. The command also uses the --stopReason option that writes the reason for the shut down to the logs.

```
$ bin/stop-server --stopTime 20120606134500Z --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --stopReason "Scheduled offline maintenance"
```

To Restart the Server

Re-start the Directory Server using the bin/stop-server command with the `--restart` or `-R` option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again.

- Go to the server root directory, and run the bin/stop-server command with the -R or --restart options.

```
$ bin/stop-server --restart
```

Run the Server as a Microsoft Windows Service

The server can run as a Windows service on Windows Server 2012 R2 and Windows Server 2016. This enables log out of a machine without the server being stopped.

To Register the Server as a Windows Service

Perform the following steps to register the server as a service:

1. Stop the server with `bin/stop-server`. A server cannot be registered while it is running.
2. Register the server as a service. From a Windows command prompt, run `bat/register-windows-service.bat`.
3. After a server is registered, start the server from the Windows Services Control Panel or with the `bat/start-server.bat` command.

   Command-line arguments for the `start-server.bat` and `stop-server.bat` scripts are not supported while the server is registered to run as a Windows service. Using a task to stop the server is also not supported.

To Run Multiple Service Instances

Only one instance of a particular service can run at one time. Services are distinguished by the `wrapper.name` property in the `<server-root>/config/wrapper-product.conf` file. To run additional service instances, change the `wrapper.name` property on each additional instance. Descriptions of the services can also be added or changed in the `wrapper-product.conf` file.

To Deregister and Uninstall Services

While a server is registered as a service, it cannot run as a non-service process or be uninstalled. Use the `bat/deregister-windows-service.bat` file to remove the service from the Windows registry. The server can then be uninstalled with the `uninstall.bat` script.

Log Files for Services

The log files are stored in `<server-root>/logs`, and filenames start with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or due to file size. Only the last three log files are retained. These configurations can be changed in the `<server-root>/config/wrapper.conf` file.

Uninstalling the Server

The Directory Server provides an `uninstall` command-line utility for quick and easy removal of the code base.

To uninstall a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode is a text-based interface that prompts the user for input. You can start the command using the `bin/uninstall` command with the --cli option. The utility prompts you for input if more data is required.
- **Non-Interactive Command-Line Mode**. Non-interactive mode suppresses progress information from being written to standard output during processing, except for fatal errors. This mode is convenient for scripting and is invoked using the `bin/uninstall` command with the --no-prompt option.

Note: For stand-alone installations with a single Directory Server instance, you can also manually remove the Directory Server by stopping the server and recursively deleting the directory and subdirectories. For example:

```
$ rm -rf /ds/PingDirectory
```

To Uninstall the Server in Interactive Mode

Interactive mode uses a text-based, command-line interface to help you remove your instance. If `uninstall` cannot remove all of the Directory Server files, the `uninstall` tool generates a message with a list of the files and directories that must be manually deleted. The `uninstall` command must be run as either the root user or the same user (or role) that installed the Directory Server.

1. From the server root directory, run the `uninstall` command.

```
$ ./uninstall --cli
```

2. Select the components to be removed. If you want to remove all components, press **Enter** to accept the default (remove all). Enter the option to specify the specific components that you want to remove.

```
Do you want to remove all components or select the components to remove?

1) Remove all components
2) Select the components to be removed

q) quit
Enter choice [1]:
```

3. For each type of server component, press **Enter** to remove them or type `no` to keep it.

```
Remove Server Libraries and Administrative Tools? (yes / no) [yes]:
Remove Database Contents? (yes / no) [yes]:
Remove Log Files? (yes / no) [yes]:
Remove Configuration and Schema Files? (yes / no) [yes]:
Remove Backup Files Contained in bak Directory? (yes / no) [yes]:
Remove LDIF Export Files Contained in ldif Directory? (yes / no) [yes]:
```

4. If the Directory Server is part of a replication topology, type `yes` to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a stand-alone server, continue to step 7.
5. Type the Global Administrator ID and password to remove the references to this server in other replicated servers. Then, type or verify the host name or IP address for the server that you are uninstalling.
6. Next, select how you want to trust the server certificate if you have set up SSL or StartTLS. For this example, press **Enter** to accept the default.

```
How do you want to trust the server certificate for the Directory Server
on server.example.com:389?

1) Automatically trust
2) Use a trust store
3) Manually validate

Enter choice [3]:
```

7. If your Directory Server is running, the server is shutdown before continuing the uninstall process. The uninstall processes the removal requests and completes. View the logs for any remaining files. Manually remove any remaining files or directories, if listed.

To Uninstall the Server in Non-Interactive Mode

The `uninstall` utility provides a non-interactive method to enter the command with the --no-prompt option. Another useful argument is the --forceOnError option that continues the uninstall process when an error is encountered. If an option is incorrectly entered or if a required option is omitted and the --forceOnError option is not used, the command will fail and abort.

1. From the server root directory, run `uninstall` tool with the --remove-all option to remove all of the Directory Server's libraries. The --quiet option suppresses output information and is optional. The following command assumes that the Directory Server is stand-alone and not part of a replication topology.

```
$ ./uninstall --cli --remove-all --no-prompt --quiet --forceOnError
```

2. If any files or directories remain, manually remove them.

To Uninstall Selected Components in Non-Interactive Mode

From the server root directory, run `uninstall` with the --backup-files option to remove the Directory Server's backup files. Use the --help or -H option to view the other options available to remove specific components.

```
$ ./uninstall --cli --backup-files --no-prompt --quiet --forceOnError
```

Chapter 4

Upgrading the Server

Topics:

Ping Identity issues software release builds periodically with new features, enhancements, and fixes for improved server performance. Administrators can use the Directory Server's update utility to upgrade the current server code version.

This chapter presents some update scenarios and their implications that you should consider when upgrading your server code.

Upgrade Overview and Considerations

The upgrade process involves downloading and unzipping a new version of the Directory Server ZIP file on the server to be updated, and running the `update` utility with the `--serverRoot` or `-R` option value from the new root server pointing to the installation to be upgraded.

Consider the following when upgrading replicating servers:

- Upgrade affects only the server being upgraded. The process does not alter the configuration of other servers.
- The `update` tool will verify that the version of Java that is installed meets the new server requirements. To simplify the process, install the version of Java that is supported by the new server before running the tool.
- To be safe, backup the user data (userRoot) before an upgrade. Restoring from a backup could be necessary if all other servers in the replication topology have been upgraded and a database or encoding change in the new server version prevents the database from being used with the older server version. The `update` and `revert-update` utilities will issue a warning when this is the case.
- Temporarily raise the replication purge delay for all servers in the topology to cover the expected downtime for maintenance. This will result in a temporary increase in disk usage for the replicationChanges database stored in `<server-root>/changelogDb`.
- Replication does not need to be disabled on a server before an upgrade.
- Make sure upgraded servers are working as expected before upgrading the last server in the topology
- Enable new features after all replicating servers are upgraded.

About Updating Servers in a Topology

An update to the current release includes significant changes, and the introduction of a topology registry, which will store information previously stored in the admin backend (server instances, instance and secret keys, server groups, and administrator user accounts). For the admin backend to be migrated, the `update` tool must be provided LDAP authentication options to the peer servers of the server being updated.

The LDAP connection security option requested (either plain, TLS, StartTLS, or SASL) must be configured on every server in the topology. The LDAP credentials must be present on every server in the topology, and must have permissions to read from the admin backend and the config backend of every server in the topology. For example, a root DN user with the `inherit-default-privileges` set to true (such as the `cn=Directory Manager` user) that exists on every server can be used.

After enabling or fixing the configuration of the LDAP connection handler(s) to support the desired connection security mechanism on each server, run the following `dsframework` command on the server being updated so that its admin backend has the most up-to-date information:

```
$ bin/dsframework set-server-properties \
  --serverID serverID \
  --set ldapport:port \
  --set ldapsport:port \
  --set startTLSEnabled:true
```

The `update` tool will verify that the following conditions are satisfied on every server in the topology before allowing the update:

- When the first server is being updated, all other servers in the topology must be online. When updating additional servers, all topology information will be obtained from one of the servers that has already been updated. The `update` tool will connect to the peer servers of the server being updated to obtain the necessary information to populate the topology registry. The provided LDAP credentials must have read permissions to the config and admin backends of the peer servers.
- The instance name is set on every server, and is unique across all servers in the topology. The instance name is a server's identifier in the topology. After all servers in the topology have been updated, each server will be uniquely identified by its instance name. Once set, the name cannot be changed. If needed, the following command can be used to set the instance name of a server prior to the update:

```
$ bin/dsconfig set-global-configuration-prop \
  --set instance-name:uniqueName
```

- The cluster-wide configuration is synchronized on all servers in the topology. Older versions have some topology configuration under `cn=cluster,cn=config` (JSON attribute and field constraints). These items did not support mirrored cluster-wide configuration data. An update should avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. To synchronize the cluster-wide configuration data across all servers in the topology, run the `config-diff` tool on each pair of servers to determine the differences, and use `dsconfig` to update each instance using the `config-diff` output. For example:

```
$ bin/config-diff --sourceHost hostName \
  --sourcePort port \
  --sourceBindDN bindDN \
  --sourceBindPassword password \
  --targetHost hostName \
  --targetPort port \
  --targetBindDN bindDN \
  --targetBindPassword password
```

If any of these conditions are not satisfied, the `update` tool will list all of the errors encountered for each server, and provide instructions on how to fix them.

To Upgrade the Directory Server

Perform an upgrade with the following steps.

1. Download and unzip the new version of the Directory Server in a location outside the existing server's installation. For these steps, assume the existing server installation is in `/prod/PingDirectory` and the new server version is unzipped into `/home/stage/PingDirectory`.
2. Run the `update` tool provided with the new server package to update the existing Directory Server. The update tool may prompt for confirmation on server configuration changes if it detects customization.

```
$ /home/staging/PingDirectory/update --serverRoot /prod/PingDirectory
```

To Upgrade the RPM Package

If the Linux RPM package was used to install the Directory Server, the following should be performed to upgrade the server.

- Assume that the new RPM package, PingDirectory-<new-version>.rpm, is placed in the server root directory. From the server root directory, run the `rpm` command with the --upgrade option.

```
$ rpm --upgrade PingDirectory-<new-version>.rpm
```

The RPM package does not support a revert option once the build is upgraded.

The upgrade history is written to /opt/ping-identity/ds/PingDirectory/PingDirectory/history/<timestamp>/update.log.

Reverting an Update

An installation can be reverted (one level back) using the `revert-update` tool. The `revert-update` tool will place the server's binaries and configuration back to its prior state. If multiple updates have been performed, the tool can be run multiple times to revert to each prior update sequentially, but only one level at a time.

1. Stop the server and run the `revert-update` command:

```
$ bin/stop-server
$ ./revert-update
```

2. Before starting the server, make sure there were no warnings, such as:

   Warning: Additional steps must be taken once this revert has completed. The current server uses an on disk database format that is not compatible with the server version being reverted to. After performing the revert, compatible databases must be put in place before restarting the server. Different steps must be taken to restore the Local DB Backend, Replication Changelog, and LDAP Changelog databases.

   This type of message indicates that the userRoot, replicationChange and changelog (if enabled) databases need to be restored before starting the server. These databases can be backed up from another server of the same version in the replication topology and restored on the newly reverted server. For example, from another server in the topology, perform the following:

```
$ bin/backup --backendID userRoot --backupDirectory /backups/userRoot
$ bin/backup --backendID replicationChanges --backupDirectory /backups/
replicationChanges
$ bin/backup --backendID changelog --backupDirectory /backups/changelog
```

   Transfer the backups to the reverted server and restore each database with the following commands:

```
$ bin/restore --backupDirectory /transfered/userRoot
$ bin/restore --backupDirectory /transfered/changelog
$ bin/restore --backupDirectory /transfered/replicationChanges
```

3. After all databases have been restored, start the server with the following command:

```
$ bin/start-server
```

Configure SCIM After Upgrade

Modifications in SCIM PATCH are mapped directly to LDAP modifications to use the matching rules configured in the Directory Server, when matching deleted values. Since the SCIM PATCH is now applied by the Directory Server, the Permissive Modify Request Control (1.2.840.113556.1.4.1413) is now required by the SCIM component. This ensures that adding an existing value or deleting a non-existent value in the PATCH request will not generate an error. This affects upgrades from server versions prior to 5.0.0.

To continue using the SCIM component after an upgrade, access controls and configuration must be updated to allow access to the Permissive Modify Request Control. Run the `dsconfig` commands to update these components:

```
$ dsconfig set-access-control-handler-prop \
  --remove 'global-aci:(targetcontrol="1.3.6.1.1.13.2 ||
 1.2.840.113556.1.4.473 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9
 || 1.3.6.1.1.12")(version 3.0;acl "Authenticated access to controls used by
 the SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473
 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9 || 1.3.6.1.1.12 ||
 1.2.840.113556.1.4.1413")(version 3.0;acl "Authenticated access to controls
 used by the SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

Chapter

5

Tuning the Server

Topics:

The PingDirectory Server's installation process automatically determines the optimal Java Virtual Machine (JVM) settings based on calculations of the machine running setup. While the majority of the default configuration and JVM settings are suitable for most deployments, it is not uncommon in high performance environments to make slight adjustments to the Directory Server's JVM settings as well as performance and resource-related configuration changes with the `dsconfig` tool.

This chapter provides guidance for tuning the Directory Server and its tools for both optimum performance with regard to throughput and disk space usage. This chapter presents the following topics:

About Minimizing Disk Access

Most critical to directory server performance is minimizing disk access. Defining a JVM heap size that can contain the entire contents of the database cache in memory is essential to minimizing read operations from disk and achieving optimal performance. It is also important to understand that the database on-disk is comprised of transaction log files, which are only appended to. After an initial database import, the size on-disk will grow by a factor of at least 25% as inactive records accumulate within the transaction logs. Therefore, during normal operation, the on-disk size of the database transaction logs do not represent the memory needed to cache the database.

Another consideration is to minimize the size of the database based on the known characteristics of your data. Minimizing the size of the database not only reduces hard disk requirements but also reduces the memory requirements for the database cache. The Directory Server has the capability to automatically compact common parent DNs, which is an example of optimizing the database size based on known characteristics of the data.

Another consideration is to consider the write load on your server and its affect on the database. While write operations will always require an associated write to disk, an environment that sustains a high load of write operations may consider tuning the background database cleaner to minimize the size of the database on disk.

Memory Allocation and Database Cache

The Directory Server's optimal performance is dependent on the proper allocation of memory to the JVM heap, the number of processor cores in the system, and the correct combination of JVM options for optimized garbage collection. The `setup` tool for the Directory Server automatically assigns the JVM options and determines the memory allocation based on the total amount of memory on the system. However, in most production deployments, additional tuning may be required to meet the performance objectives for your system.

Most often, directory server performance tuning can be accomplished by adjusting a few settings. Tuning these settings, which include both JVM and configuration options, require an understanding of the JVM heap structure as well as the expected database usage. This section describes the basic components of the Directory Server footprint and logic behind the automated tuning of the `setup` tool.

Directory Server Process Memory

The Directory Server is comprised mostly of a JVM heap and a marginal amount of memory allocated by the JVM's execution of native code. While we frequently refer to the JVM Heap as the maximum memory consumed by the Directory Server, the actual process size will be slightly larger than the Xmx value due to accumulation of small chunks of native code that Java requires for things, such as SSL sockets.

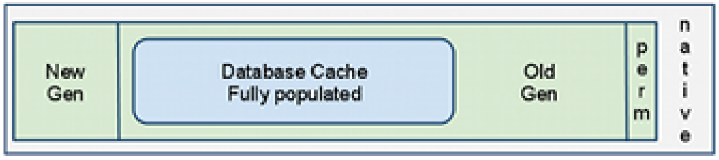


Figure 1: JDK Heap Structure

Within the JVM Heap, the principal memory components are the New and Old Generations. The New Generation is a smaller area of memory where all data is initially allocated and is cleaned of garbage often. Any data that stays "alive" long enough will be promoted to the Old Generation for the longer term. The Old Generation is where the database cache will eventually reside. The Old Generation size is computed from the leftover heap after defining the MaxHeapSize and New Generation sizes; therefore, it is not explicitly stated in the JVM options. A typical set of Generation definitions for the JVM is as follows, where mx and ms values represent the heap size:

```
-Xmx16g -Xms16g -XX:MaxNewSize=2g -XX-NewSize=2g
```

Note: The mx and ms values should always be the same, and the MaxNewSize/NewSize values should be the same. This will help avoid negative changes in performance.

The MaxNewSize/NewSize values should never need to exceed 2g. The `setup` and `dsjavaproperties` tools set MaxNewSize/NewSize values based on the results of extensive performance testing, and should not need to be changed.

A Method for Determining Heap and Database Cache Size

The most straightforward approach to defining the proper memory allocation of the directory server components is to use the Directory Server `setup` command on hardware that represents the target production platform, especially with regard to process and memory, and the largest heap size that the `setup` tool will allow. After running `setup`, any schema and production database settings should be defined in preparation for the database import using the `import-ldif` tool.

Note: At the moment after an `import-ldif`, the database is at its most optimized state on disk with no inactive records. Over time, the on-disk representation of the database will grow as much as 25-50% as inactive records accumulate before being removed by the server’s cleaner thread.

After the database is imported, the server should be started and a configuration change made to the backend. In this scenario, set the *prime-method* to "preload" on the userRoot backend configuration. Once the change is made, re-start the Directory Server and watch for a successful preload message at the end of startup. If preloading did not complete, the server should be stopped. The `start-server.java-args` entry in the `config/java.properties` file should be edited to use larger values for `-Xmx` and `-Xms` arguments. Then run the `bin/dsjavaproperties` command and restart the server. If preloading completed successfully, the database cache utilization percentage will be of interest. The `status` command will display something like the following:

```
             --- JE Environment ---
ID    : Cache Full : Cache  : On-Disk : Alert
---------:------------:--------:---------:------
userRoot : 30%     : 1.1 gb : 868.6mb : None
```

Looking at the above output and knowing that the database is fully loaded into cache, the 30% utilization is comfortable cushion for future database growth. In general, it is best to leave at least 10-20% cache headroom available.

During this scenario, it was clear from the `start-server` output that the database primed completely and our interpretation of the `status` output was sound. To see the state of the database cache with more detail, perform an `ldapsearch` on the backend monitor.

In addition to the user configured backends, there may be backends for replication and changelog. The heap is shared among all backends. The amount allocated to each backed is calculated according to the procedure in the next section.

Automatic DB Cache Percentages

The setup process automatically tunes the percentage of the `db-cache-percent` property for the userRoot backend based on the maximum configured JVM heap size. This is only done for the userRoot backend during setup. Other backends created by the user are allocated 10%. The allocation can be changed if needed. When setting up the server, perform the following steps:

- Install the server with necessary memory. The server will autotune the size of the cache.
- Set the autotuned cache size to the limit for the combined cache sizes of all of the backends.
- Divide the server cache based on the expected size of the data in each backend.

Automatic Memory Allocation

If the Memory Tuning feature is enabled during setup, the `setup` algorithm determines the maximum JVM heap size based on the total amount of available system memory. If Memory Tuning is not selected, the server allocates a maximum JVM heap of 384 MB. The Directory Server also allows you to specify the maximum heap size during the setup process. You can enable Memory Tuning during the setup process by selecting the feature during the interactive command-line mode, adding the --jvmTuningParameter option using the `setup` tool in non-interactive command-line mode, or regenerating the java properties file with `bin/dsjavaproperties` and the --jvmTuningParameter options (seen in JVM Properties for Server and Command-Line Tools page 75).

If Memory Tuning is selected, the server can allocate the maximum JVM heap depending on the total system memory. The following table displays the automatically allocated maximum JVM heap memory based on available system memory.

Table 2: Allocated Max JVM Memory if Tuning is Enabled

Available Memory	Allocated JVM Memory
16 GB or more using a 64-bit JVM	The maximum JVM heap size will be set to 70% of total system memory. If the maximum JVM heap size is less than or equal to 128GB of memory (which should be the case for systems with up to 160 GB of memory), then the initial heap size will be set to equal the maximum heap size.
6 GB–16 GB using a 64-bit JVM	total system memory - 4 GB
4 GB–6 GB using a 64-bit JVM	2 GB
2 GB–4 GB	512 MB
1 GB–2 GB	384 MB

Automatic Memory Allocation for the Command-Line Tools

At setup, the Directory Server automatically allocates memory to each command-line utility based upon the maximum JVM heap size. The server sets each command-line utility in the `config/java.properties` with -Xmx/Xms values depending on the expected memory needs of the tools. Because some tools can be invoked as a server task while the server is online, there are two definitions of the tool in the `config/java.properties` file: one with .online and one with .offline added to the name. The online invocations of the tools typically require minimal memory as the task is performed within the Directory Server's JVM. The offline invocations of the tools, for example, `import-ldif.offline` and `rebuild-index.offline`, can require the same amount of memory that is needed by the Directory Server.

Beyond the offline tool invocations, some tools, such as `ldap-diff` and `verify-index`, may need more than the minimal memory if large databases are involved. The table below lists the tools that are expected to have more than the minimal memory needs along with the rules for defining the default heap size.

Table 3: Default Memory Allocation to the Command-Line Tools

Command-Line Tools	Allocated JVM Memory
start-server, import-ldif (offline), rebuild-index (offline)	MaxHeapSize
backup (offline), dbtest export-ldif (offline), ldap-diff, restore (offline), scramble-ldif, summarize-access-log, verify-index	If Max System Memory is: Greater than or equal to 16 GB: set Heap to 3 GB Greater than or equal to 8 GB: set Heap to 1 GB Greater than or equal to 4 GB: set Heap to 512 MB Under 4 GB: set Heap to 256 MB

Database Preloading

Key to Directory Server performance is the ability to maintain the database contents in the database cache within the JVM memory. With a properly sized database cache, a priming method of "preload" directs the server to load the database contents into memory at server startup before accepting the first client connection. The time needed to preload the database is proportional to the database size. To avoid priming, the server can be started with the `start-`

`server --skipPrime` command. If the priming method is none or the `--skipPrime` option is specified at startup, the database cache will slowly build as entries are accessed. This could take several days to reach optimal performance.

The "preload" priming method is suitable for nearly all Directory Server deployments. If the size of the database precludes storing the whole database in memory, there are priming alternatives for optimizing server performance. This type of deployment is considered disk-bound since the disk is accessed when processing most operations. See the section Disk-Bound Deployments for more information. The remaining priming options are applicable to these environments.

The Directory Server database `prime-method` property configures how the caches get primed, what gets primed (data, internal nodes, system indexes) and where it gets primed (database cache, filesystem cache, or both). The `prime-method` property is a multi-valued option that enables preloading the internal nodes into the database cache before the server starts, and then primes the values in the background by cursoring across the database. For more details, see the PingDirectory Server Configuration Reference.

The following is a summary of the priming methods:

- **Preload All Data**. Prime the contents of the backend into the database cache.
- **Preload Internal Nodes Only**. Prime only internal database structure information into the database cache, but do not prime any actual data. (This corresponds to the cache-keys-only cache-mode.)
- **Cursor Across Indexes**. Use the `cursor-across-indexes` property to iterate through backend contents. This is similar to (and may be slower than) using the preload mechanism, but it enables priming to happen in the background after the server has started. This is used when shorter start up times are desired, and the slower performance of an uncached database is acceptable until the database is primed.

Configuring Database Preloading

Use the `dsconfig` tool to set the database priming method. If multiple prime methods are used, the order in which they are specified in the configuration is the order in which they will be performed. Changing the preloading option requires re-starting the Directory Server. The following procedure shows how to configure database preloading.

To Configure Database Preloading

1. Set the prime method to "preload" to load the database contents from disk into memory when the server starts up. This eliminates the need for the server to gradually prime the database cache using client traffic, and ensures that the server has optimal performance when it starts to receive client connections.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --set prime-method:preload
```

2. Re-start the Directory Server to apply the changes using `bin/stop-server` and then, `bin/ start-server`.

To Configure Multiple Preloading Methods

1. To achieve the benefits of preloading without delaying server startup, `prime-method` can be set to `preload-internal-nodes-only`, which caches all of the keys within the database but not the values. The database values themselves can be cached in the background once the server has been started with the `cursor-across-indexes` option.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --add prime-method:preload-internal-nodes-only \
  --add prime-method:cursor-across-indexes \
  --set background-prime:true
```

2. Re-start the Directory Server to apply the changes using `bin/stop-server` and then, `bin/start-server`.

To Configure System Index Preloading

1. Some environments have many indexes configured, though only a few are used for performance-sensitive traffic. In this case, server start up time can be reduced by only preloading the necessary indexes into the database at startup.

   ```
   $ bin/dsconfig set-backend-prop --backend-name userRoot \
     --set prime-method:preload \
     --set prime-all-indexes:false \
     --set system-index-to-prime:dn2id \
     --set system-index-to-prime:id2entry
   ```

   ```
   $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
     --index-name mail \
     --set prime-index:true
   ```

   ```
   $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
     --index-name uid \
     --set prime-index:true
   ```

   ```
   $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
     --index-name entryUUID \
     --set prime-index:true
   ```
2. Restart the Directory Server to apply the changes using `bin/stop-server` and then, `bin/start-server`.

Databases on Storage Area Networks, Network-Attached Storage, or running in Virtualized Environments

There are several considerations when using network-based storage or storage abstracted by virtualization that are not issues when databases are stored on local disks. A data durability problem occurs when remote storage or the virtualization environment experiences service interruptions, ranging from connectivity loss to total failure from power loss. Data corruption can occur when the storage layer accepts data for writing that is not made durable before a crash occurs. In these cases, a database property can be set that reduces the likelihood of data loss and data corruption. The database property `database-on-virtualized-or-network-storage` can be set on a per-backend environment basis to request all database writes to be written durably to the underlying storage.

There is a performance penalty when enabling this property, and in most cases, is not recommended except where network storage is unreliable. For network file systems, the benefits of faster recovery and less likelihood of data loss from unplanned events may outweigh the penalty. The exact overhead of enabling `database-on-virtualized-or-network-storage` will depend on the characteristics of the database, the host filesystem, storage array configuration, and network and virtualization input and output parameters. The write overhead penalty may be substantial for SAN environments. Incremental and full backup strategies should be used instead if performance is unacceptable.

To enable `database-on-virtualized-or-network-storage` for each applicable backend, use the following command as an example, which references the configuration for the userRoot backend:

```
$ bin/dsconfig set-global-configuration-prop \
  --set database-on-virtualized-or-network-storage:true
```

This should be set to `false` if the database is on a local disk.

Database Cleaner

Production environments that have a high volume of write operations may require cleaner thread tuning to control the on-disk database size as log files with inactive nodes wait to be cleaned and deleted. The Directory Server stores its Oracle® Berkeley DB Java Edition (JE) database files on-disk in the `db` directory. Each JE database log file is labelled nnnnnnnn.jdb, where nnnnnnnn is an 8-digit hexadecimal number that starts at 00000000 and is increased by 1 for each file written to disk. JE only appends data to the end of each file and does not overwrite any existing data. JE uses one or more cleaner threads that run in the background to compact the number of JE database (db) files.

The cleaner threads begin by scanning the records in each db file, starting with the file that contains the smallest number of active records. Next, the cleaner threads append any active records to the most recent database file. If a record is no longer active due to modifications or deletions, the cleaner threads leave it untouched. After the db file no longer has active records, the cleaner threads can either delete the file or rename the discarded file. Note that because of this approach to cleaning, the database size on-disk can temporarily increase when cleaning is being performed and files are waiting to be removed.

The Local DB Backend configuration object has two properties that control database cleaning: db-cleaner-min-utilization and db-num-cleaner-threads. The db-cleaner-min-utilization property determines, by percentage, when to begin cleaning out inactive records from the database files. By default, the property is set to 75, which indicates that database cleaning ensures that at least 75% of the total log file space is devoted to live data. Note that this property only affects the on-disk representation of the database and not the in-memory database cache—only live data is ever cached in memory.

The db-num-cleaner-threads property determines how many threads are configured for db cleaning. The default single cleaner thread is normally sufficient. However, environments with a high volume of write traffic may need to increase this value to ensure that database cleaning can keep up.

If the number of database files grow beyond your expected guidelines or if the Directory Server is experiencing an increased number of update requests, you can increase the number of cleaner threads using the dsconfig tool (select **Backend** > **select advanced properties** > **db-num-cleaner-threads**).

Compacting Common Parent DNs

The PingDirectory Server compacts entry DNs by tokenizing common parent DNs. Tokenizing the common parent DNs allows you to increase space usage efficiency when encoding entries for storage. The Directory Server automatically defines tokens for base DNs for the backend (for example, dc=example,dc=com). You can also define additional common base DNs that you want to tokenize. For example, use the following configuration to tokenize two branches, ou=people,dc=example,dc=com and ou=customers,dc=example,dc=com:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --add "compact-common-parent-dn:ou=people,dc=example,dc=com" \
  --add "compact-common-parent-dn:ou=customers,dc=example,dc=com"
```

Import Thread Count

For most systems, the default setting of 16 threads is sufficient and provides good import performance. On some systems, increasing the import thread count may lead to improved import performance, while selecting a value that is too large can actually cause import performance to degrade. If minimizing LDIF import time is crucial to your deployment, you must determine the optimal number of import threads for your system, which is dependent on both the underlying system and the dataset being imported.

You can use the dsconfig command to set the number of import threads as follows:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot --set import-thread:24
```

JVM Properties for Server and Command-Line Tools

The Directory Server and tools refer to the config/java.properties file for JVM options that include important memory settings. The java.properties file sets the default Java arguments for the Directory Server and each command-line utility including the default *JAVA_HOME* path.

The java.properties is generated at server setup time and defines memory-related JVM settings based on the user-provided value for max heap size if aggressive memory tuning option was selected at setup. Most of the JVM options specified for both server and tools do not need customization after setup. The exception is the -Xmx/Xms options, which specify the maximum and initial JVM heap size. See the section on Memory Allocation and Database Cache for advice on tailoring the -Xmx/Xms values.

Other than altering the heap size of the server process (start-server) or command-line tools, the most common change required to java.properties is when it is desired to update the JVM version. A single edit will apply the new JVM to all server and tool use.

Applying Changes Using dsjavaproperties

To apply the changes to the config/java.properties file, edit the file manually, and then run the bin/dsjavaproperties utility. The dsjavaproperties tool uses the information contained in the config/java.properties file to generate a lib/set-java-home script (or lib\set-java-home.bat on Microsoft Windows systems), which is used by the Directory Server and all of its supporting tools to identify the Java environment and its JVM settings. During the process, dsjavaproperties calculates an MD5 digest of the contents of the config/java.properties file and stores the digest in the generated set-java-home script.

The dsjavaproperties utility also performs some minimal validation whenever the property references a valid Java installation by verifying that $(java-home)/bin/java exists and is executable.

If you make any changes to the config/java.properties file but forget to run bin/dsjavaproperties, the Directory Server compares the MD5 digest with the version stored in set-java-home and sends a message to standard error if the digests differ:

```
WARNING -- File /ds/PingDirectory/config/java.properties has been edited
 without
running dsjavaproperties to apply the changes
```

To Update the Java Version in the Properties File

To change the version of java that is used by the server and tools, it is necessary to edit the config/java.properties file and apply the change by invoking bin/dsjavaproperties with no command line options. Also, the server must be restarted for the change to take affect.

Inside config/java.properties, alter the value of default.java-home to point to the java correct JRE. Any time the config/java.properties file is updated, the bin/dsjavaproperties tool must be run to apply the new configuration.

```
$ bin/dsjavaproperties
```

To Regenerate the Java Properties File

The dsjavaproperties command provides a --initialize option that allows you to regenerate the Java Properties file specifically if you set up the Directory Server using standard memory usage but opt for aggressive memory tuning after setup. Rather than reconfigure the Java Properties file by re-running setup or manually editing the java.properties file, you can regenerate the properties file for aggressive memory tuning. Any existing file will be renamed with a ".old" suffix.

- Run the dsjavaproperties command to regenerate the java properties file for aggressive memory tuning:

  ```
  $ bin/dsjavaproperties --initialize --jvmTuningParameter AGGRESSIVE
  ```

JVM Garbage Collection Using CMS

To ensure reliable server performance with Java, the Directory Server depends on Java's Concurrent Mark and Sweep process (CMS) for background garbage collection. There are several garbage collection options, with CMS being the ideal choice for consistent system availability. The CMS collector runs as one or more background threads, for the most part, within the JVM, freeing up space in JVM Heap from an area called the Old Generation.

One of the criteria used by CMS to determine when to start background garbage collection is a parameter called `CMSInitiatingOccupancyFraction`. This percentage value, which applies to the Old Generation, is a recommendation for the JVM to initiate CMS when data occupancy in Old Generation reaches the threshold.

To understand this CMS property, it is important to know how large the Old Generation is and how much data in the Old Generation is expected to be occupied by the database cache. Ideally, the database cache takes less than 70% of the space available in the Old Generation, and the `CMSInitiatingOccupancyFraction` value of 80 leaves plenty of headroom to prevent the JVM from running out of space in Old Generation due to an inability for CMS to keep up. Because CMS takes processing resources away from the Directory Server, it is not recommended to set the `CMSInitiatingOccupancyFraction` at or below the expected database cache size, which would result in the constant running of CMS in the background. See the section on Memory Footprint and Database Cache for a description of determining Old Generation size.

When the CMS collection process cannot keep pace with memory demands in the Old Generation, the JVM will resort to pausing all application processing to allow a full garbage collection. This event, referred to as a *stop-the-world* pause, does not break existing TCP connections or alter the execution of the Directory Server requests. The goal in tuning CMS is to prevent the occurrence of these pauses. When one does occur, the Directory Server will generate an alert, after the pause, and record the pause time in the error log.

Because determining an ideal `CMSInitiatingOccupancyFraction` can be difficult, the approach we have taken is to warn if the Directory Server detects a garbage collection pause by generating a recommended value for the occupancy threshold based on the current amount of memory being consumed by the backend caches. Unfortunately, it is not possible for an administrator to determine the ideal occupancy threshold value in advance. Therefore, to warn of any impending garbage collection pauses, the Directory Server calculates a recommended value for the `CMSInitiatingOccupancyFraction` property and exposes it in the JVM Memory Usage monitor entry in the following attribute:

```
recommended-cms-initiating-occupancy-fraction-for-current-data-set
```

Also, when you start the server, you will see an administrative alert indicating the current state of the `CMSInitiatingOccupancyFraction` and its recommended value.

```
$ bin/start-server [20/April/2012:10:35:25 -0500] category=CORE
 severity=NOTICE msgID=458886
msg="PingDirectory Server 7.0.0.0 (build 20120418135933Z, R6226) starting up"

... (more output) ...

[20/April/2012:10:35:53 -0500] category=UBID_EXTENSIONS severity=NOTICE
msgID=1880555580 msg="Memory-intensive Directory Server
components are configured to consume 71750382 bytes of memory:
['userRoot local DB backend' currently consumes 26991632 bytes and
can grow to a maximum of 64323584 bytes, 'changelog cn=changelog backend'
currently consumes 232204 bytes and can grow to a maximum of 2426798 bytes,
'Replication Changelog Database' currently consumes 376661 bytes and can
grow to a maximum of 5000000 bytes]. The configured value of
CMSInitiatingOccupancyFraction is 36 which is less than the minimum
recommended value (43) for the server's current configuration. Having
this value too low can cause the Concurrent Mark and Sweep garbage
collector to run too often, which can cause a degradation of throughput
and response time. Consider increasing the CMSInitiatingOccupancyFraction
value to at least the minimum value, preferably setting it to the
recommended value of 55 by editing the config/java.properties file,
running dsjavaproperties, and restarting the Directory Server.
If the server later detects that this setting actually leads to a
performance degradation, a separate warning message will be logged.
If this server has not yet been fully loaded with data, then you
can disregard this message"

[20/April/2012:10:35:53 -0500] category=CORE severity=NOTICE msgID=458887
msg="The Directory Server has started successfully"
```

The Directory Server only makes a recommendation if all of the backends are preloaded and the `CMSInitiatingOccupancyFraction` JVM property is explicitly set, which is done automatically. For example, if you installed the Directory Server and specified that the database be preloaded (or "primed") at startup, then the Directory Server can make a good recommendation for the Directory Server when a pause occurs. If the backend database cache is not full and has not been preloaded, then the recommended value may be an inaccurately low value.

Note: The generated value for the Directory Server property could change over time with each Directory Server build, Java release, or changes in data set. If the current value is fairly close to the recommended value, then there is no need to change the property unless the server experiences a JVM pause.

If the Directory Server experiences a JVM garbage collection pause, you can retrieve the recommended value from the server, reset the Directory Server property, run `dsjavaproperties`, and restart the server.

To Determine the CMSInitiatingOccupanyFraction

1. If you set the Preload Database at startup option during the installation, then skip to step 3. If you are not sure, retrieve the `prime-method` property for the backend as follows:

```
$ bin/dsconfig get-backend-prop --backend-name userRoot \
  --property prime-method
```

2. If the `prime-method` property was not configured, use `bin/dsconfig` to set the property to PRELOAD, and then, restart the Directory Server to preload the database cache.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set prime-method:preload

$ bin/stop-server
$ bin/start-server
```

3. At startup, you will see an administrative message if the current `CMSInitiatingOccupancyFraction` property is below the recommended value. You can get the recommended value from this message and change it in the `config/java.properties` file in step 5.
4. If you were unable to see the recommended `CMSInitiatingOccupancyFraction` property at startup presented in the previous step, first you must pre-tune the value of the `CMSInitiatingOccupancyFraction` property to ensure that all of the data is imported into the server and preloading is enabled in the backend. Next, retrieve the recommended `CMSInitiatingOccupancyFraction` value by issuing the following search. If the `recommended-cms-initiating-occupancy-fraction-for-current-data-set` is not present, then make sure that the server has been restarted since enabling preload for the backend(s).

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectclass=ds-memory-usage-monitor-entry)" \
  cms-initiating-occupancy-fraction \
  recommended-cms-initiating-occupancy-fraction-for-current-data-set
```

```
dn: cn=JVM Memory Usage,cn=monitor
cms-initiating-occupancy-fraction:80
recommended-cms-initiating-occupancy-fraction-for-current-data-set:55
```

5. Open the `config/java.properties` file using a text editor, manually edit the `CMSInitiatingOccupancyFraction` or any other property to its recommended value in the `start-server.java-args` property, and then, save the file when finished. (The following arguments are recommended for a Sun 5440 server. Contact your authorized support provider for specific assistance.):

```
start-server.java-args=-d64 -server -Xmx20g -Xms20g -XX:MaxNewSize=1g
-XX:NewSize=1g -XXParallelGCThreads=16 -XX:+UseConcMarkSweepGC
-XX:+CMSConcurrentMTEnabled -XX:+CMSParallelRemarkEnabled
-XX:+CMSParallelSurvivorRemarkEnabled -XX:ParallelCMSThreads=8
-XX:CMSMaxAbortablePrecleanTime=3600000
-XX:+CMSScavengeBeforeRemark -XX:RefDiscoveryPolicy=1
```

```
-XX:CMSInitiatingOccupancyFraction=55 -XX:+UseParNewGC
-XX:+UseBiasedLocking -XX:+UseLargePages
-XX:+HeapDumpOnOutOfMemoryError
```

The `-XX:ParallelGCThreads` should be limited to 16 (default) or to 8 for smaller systems. Also, the `-XX:ParallelCMSThreads` should be limited to 8.

6. Run the `bin/dsjavaproperties` command to apply the changes.

```
$ bin/dsjavaproperties
```

7. Restart the Directory Server.

Tuning For Disk-Bound Deployments

For best performance, configure the Directory Server to fully cache the DIT in the backend database cache. Directory Server configuration assumes this scenario. For databases too large to fit in memory, other options are available:

- Configure the server for a disk-bound data set (when the database is stored on an SSD, this configuration yields server performance that is comparable to a fully-cached scenario).
- Use uncached attributes and/or entries as described in the following section.
- Use a Directory Proxy Server in an entry-balancing deployment, which allows all data to be cached in a partitioned environment.

To Tune for Disk-Bound Deployments

To configure the server for a disk-bound configuration, follow these steps:

1. When installing the server, choose the "aggressive" option for JVM memory configuration and to preload the data when the server starts.
2. Set the `default-cache-mode` of the `userRoot` backend to `cache-keys-only`.
3. Set operating system `vm.swappiness` to 0 to protect the Directory Server JVM process from an overly aggressive filesystem cache.
4. When the data set is imported with the above settings, verify in the `import-ldif` output that the cached portions of the data set fit comfortably within the database cache.

Uncached Attributes and Entries

Although achieving optimal Directory Server performance requires that the entire data set be fully cached, there may be deployments in which fully caching the data set is not possible due to hardware or financial constraints, or in which acceptable performance can be achieved by only caching a portion of the data. The Directory Server already provides support for controlling caching on a per-database basis (e.g., to cache only certain indexes and/or system databases), but these features may not provide sufficient control over how memory is used, particularly with regard to which entries are included in the cache, and they do not provide any degree of control over caching only a portion of attributes.

To better address the needs of environments that require partial caching, the Directory Server provides two new options: the ability to exclude certain *entries* from the cache, and the ability to exclude certain *attributes* from the cache. The Directory Server uses an `uncached-id2entry` database container, which is similar to the `id2entry` database that maps an entry's unique identifier and its encoded representation. The `uncached-id2entry` database contains either complete and/or partial representations of entries that are intended to receive less memory for caching. For example, if an entry has a particulary large attribute and the system has hardware constraints on memory, then you can configure the system to not cache this particular attribute or entry. This functionality is only available for the local DB backend, which uses the Berkeley DB Java Edition database.

The `uncached-id2entry` database can be included in the set of databases to prime, but if priming is to be performed, it will only include internal nodes and not leaf nodes. For example, the internal nodes of the `uncached-`

id2entry database will be included in the preload if the prime-all-indexes option is set to "true," or if the system-index-to-prime-internal-nodes-only option has a value of "uncached-id2entry".

Backup/Restore. There are no special considerations for backup and restore with regard to uncached entries and attributes. Backup will successfully save your database contents including uncached entries and attributes. Because of the way the server deals with changes to uncached entry and uncached attribute configuration, there is no problem with restoring a backup that was taken with a different uncached entry configuration than is currently in place for the server. Any entries encoded in a manner that is inconsistent with the current uncached entry or uncached attribute configuration will be properly re-encoded whenever they are updated, or whenever the re-encode entries task is invoked.

Replication. Replication does not propagate information about which portions of entries may have been cached or uncached, nor does it require that different replicas have the same uncached attribute or uncached entry configuration.

LDIF Import/Export. When LDIF content is imported into the server, the uncached attribute and uncached entry configuration is used to determine on a per-entry basis whether some or all of the content for that entry should be written into the uncached-id2entry database. The determination is based on the current configuration and is completely independent of and unaware of the configuration that may have been in place when the LDIF data was initially exported. Neither the LDIF import nor export tools provide any options that specifically target only cached or only uncached content, but these tools do provide the ability to include or exclude entries using search filters, or to include or exclude specific attributes.

Server Access Log. Server access log messages may include uncachedDataAcessed=true in the result message for any operation in which it was necessary to access uncached data in the course of processing the associated request. For add, delete, modify, or modify DN result messages, uncachedDataAcessed=true indicates that at least a portion of the new or updated entry was written into the uncached-id2entry database, or that at least a portion of the updated entry was formerly contained in the uncached-id2entry database. For compare result messages, it indicates that at least a portion of the target entry was contained in the uncached-id2entry database and that data from the uncached portion of the entry was required to evaluate the assertion. For search result messages, it indicates that one or more of the entries evaluated as potential matches contained uncached data, and that data from the uncached portion of at least one entry was needed in determining what data should be returned to the client.

Uncached Entry/Attribute Properties. The Directory Server provides three new advanced properties on the Local DB Backend to control the caching mode for the uncached-id2entry database:

- **uncached-id2entry-cache-mode**. Specifies the cache mode that is used when accessing the records in the uncached-id2entry database. If the system has enough memory available to fully cache the internal nodes for this database, then cache-keys-only is recommended, otherwise it is better to select no-caching to minimize the amount of memory required for interacting with the uncached-id2entry database. For more information, see the *PingDirectory Server Configuration Reference*.
- **uncached-attribute-criteria**. Specifies the criteria used to identify attributes that are written into the uncached-id2entry database, rather than the id2entry database. This property is only used for entries in which the associated uncached-entry-criteria does not indicate that the entire entry should be uncached. The property applies to all entry writes, including add, soft delete, modify, and modify DN operations, as well as LDIF import and re-encode processing. Any changes to the property take effect immediately for writes occurring after the change is made. If no value is specified, then all attributes are written into the id2entry database.
- **uncached-entry-criteria**. Specifies the criteria used to identify entries that are written into the uncached-id2entry database, rather than the id2entry database. The property applies to all entry writes, including add, soft delete, modify, and modify DN operations, as well as LDIF import and re-encode processing. Any changes to the property take effect immediately for writes occurring after the change is made. If no value is specified, then all entries are written into the id2entry database.

To Configure Uncached Attributes and Entries

The following procedure assumes that the uncached-id2entry-cache-mode property is set to the default value, cache-keys-only. For more information on the uncached-id2entry cache modes, see the *PingDirectory Server Configuration Reference*.

1. Run `dsconfig` to uncache entries that match the criteria. Here, the filter will uncache all entries that have its location set to "austin" (i.e., `l=austin`).

   ```
   $ bin/dsconfig create-uncached-entry-criteria \
     --criteria-name "Fully Uncached l=austin" --type filter-based \
     --set enabled:true --set "filter:(l=austin)"
   ```
2. Run `dsconfig` to uncache attributes that match the criteria (`attribute-type: jpegPhoto`). The `--type simple` option indicates that the simple uncached attribute criteria be used to specify the attribute-type that should be uncached, which in this example is `jpegPhoto`. For those entries that are fully stored in the `uncached-id2entry` database container, the uncached attribute will be ignored.

   ```
   $ bin/dsconfig create-uncached-attribute-criteria \
     --criteria-name "Uncached jpegPhoto" --type simple \
     --set enabled:true --set attribute-type:jpegPhoto
   ```
3. Set the uncached properties for the `userRoot` backend.

   ```
   $ bin/dsconfig set-backend-prop \
     --backend-name userRoot \
     --set "uncached-entry-criteria:Fully Uncached l=austin" \
     --set "uncached-attribute-criteria:Uncached jpegPhoto"
   ```
4. Run the `re-encode-entries` tool to initiate a task that causes a local DB `userRoot` backend to re-encode all or a specified subset of the entries that it contains. The tool does not alter the entries themselves but provides a useful mechanism for applying significant changes to the way that entries are stored in the backend. The following command initiates a task that re-encodes all fully-cached entries in the `userRoot` backend, rate-limited to no more than 100 entries per second.

   ```
   $ bin/re-encode-entries --hostname directory.example.com --port 389 \
     --bindDN uid=admin,dc=example,dc=com --bindPassword password \
     --backendID userRoot --skipFullyUncachedEntries \
     --skipPartiallyUncachedEntries --ratePerSecond 100
   ```

Chapter

6

Configuring the Server

Topics:

The out-of-the-box, initial configuration settings for the PingDirectory Server provide an excellent starting point for most general-purpose Directory Server applications. However, additional tuning might be necessary to meet the performance, hardware, operating system, and memory requirements for your production environment.

The Directory Server stores its configuration settings in an LDIF file, `config/config.ldif`. Rather than editing the file directly, the Directory Server provides command-line and an Administrative Console to configure the server. The Directory Server also includes tools to create server groups, so that configuration changes can be applied to multiple servers at one time.

This chapter presents the following topics:

About the Configuration Tools

The PingDirectory Server configuration can be accessed and modified in the following ways:

- **Using the Administrative Console**. The PingDirectory Server provides an Administrative Console for graphical server management and monitoring. The console provides equivalent functionality as the `dsconfig` command for viewing or editing configurations. All configuration changes using this tool are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands should you need to back out of a configuration.
- **Using the dsconfig Command-Line Tool**. The `dsconfig` tool is a text-based menu-driven interface to the underlying configuration. The tool runs the configuration using three operational modes: interactive command-line mode, non-interactive command-line mode, and batch mode. All configuration changes made using this tool are recorded in `logs/config-audit.log`.

About dsconfig Configuration Tool

The `dsconfig` tool is the text-based management tool used to configure the underlying Directory Server configuration. The tool has three operational modes: interactive mode, non-interactive mode, and batch mode.

The `dsconfig` tool also offers an offline mode using the --offline option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Using dsconfig in Interactive Command-Line Mode

In interactive mode, the `dsconfig` tool offers a filtering mechanism that only displays the most common configuration elements. The user can specify that more expert level objects and configuration properties be shown using the menu system.

Running `dsconfig` in interactive command-line mode provides a user-friendly, menu-driven interface for accessing and configuring the PingDirectory Server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` script without any arguments. You will be prompted for connection and authentication information to the Directory Server, and then a menu will be displayed of the available operation types.

In some cases, a default value will be provided in square brackets. For example, [389] indicates that the default value for that field is port 389. You can press **Enter** to accept the default. To skip the connection and authentication prompts, provide this information using the command-line options of `dsconfig`.

To Configure the Server Using dsconfig Interactive Mode

1. Launch the `dsconfig` tool in interactive command-line mode.

   ```
   $ bin/dsconfig
   ```
2. Next, enter the LDAP connection parameters. Enter the Directory Server host name or IP address, or press **Enter** to accept the default.
3. Enter the number corresponding to the type of LDAP connection (1 for LDAP, 2 for SSL, 3 for StartTLS) that you are using on the Directory Server, or press **Enter** to accept the default (1).
4. Next, type the LDAP listener port number, or accept the default port. The default port is the port number of the server local to the tool.
5. Enter the user bind DN (default, `cn=Directory Manager`) and the bind DN password.
6. On the **Directory Server Configuration Console** main menu, type a number corresponding to the configuration that you want to change. Note that the number can change between releases or within the same release, depending on the options selected (for example, in cases where more expert level objects and and properties are displayed).

In this example, select the number for Backend. Then, set the `db-cache-percent` to 40%. The optimal cache percentage depends on your system performance objectives and must be tuned as determined through analysis. In many cases, the default value chosen by the `setup` utility is sufficient.

7. On the **Backend management** menu, enter the number corresponding to view and edit an existing backend.
8. Select the backend to work with. In this example, using the basic object menu, only one backend that can be viewed in the directory, userRoot. Press **Enter** to accept the default.
9. From the **Local DB Backend** properties menu, type the number corresponding to the `db-cache-percent` property.
10. Enter the option to change the value, and then type the value for the `db-cache-percent` property. In this example, type `40` for "40 %“.
11. Review the changes, and then type `f` to apply them.

    Before you apply the change, the `dsconfig` interactive command-line mode provides an option to view the equivalent non-interactive command based on your menu selections. This is useful in building `dsconfig` script files for configuring servers in non-interactive or batch mode. If you want to view the equivalent `dsconfig` non-interactive command, type `d`. For more information, see Getting the Equivalent dsconfig Non-Interactive Mode Command.
12. In the **Backend management** menu, type `q` to quit the `dsconfig` tool.

To View dsconfig Advanced Properties

For most configuration settings, some properties are more likely to be modified than others. The `dsconfig` interactive mode provides an option that hides or shows additional advanced properties that administrators might want to configure.

1. Repeat steps 1–9 in the previous section using `dsconfig` in Interactive Command-Line Mode.
2. From the **Local DB Backend properties** menu, type `a` to display the advanced properties, which toggles any hidden properties.

Using dsconfig Interactive Mode: Viewing Object Menus

Because some configuration objects are more likely to be modified than others, the PingDirectory Server provides four different object menus that hide or expose configuration objects to the user. The purpose of object levels is to simply present only those properties that an administrator will likely use. The Object type is a convenience feature designed to unclutter menu readability.

The following object menus are available:

- **Basic**. Only includes the components that are expected to be configured most frequently.
- **Standard**. Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.
- **Advanced**. Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be potentially harmful if configured incorrectly.
- **Expert**. Includes all components in the Basic, Standard, and Advanced menus plus other components that should almost never require configuration or that could seriously impact the functionality of the server if not properly configured.

To Change the dsconfig Object Menu

1. Repeat steps 1–6 in the section using `dsconfig` in To Install the Directory Server in Interactive Mode.
2. On the **PingDirectory Server configuration** main menu, type **o** (letter “o”) to change the object level. By default, Basic objects are displayed.
3. Enter a number corresponding to a object level of your choice: `1` for Basic, `2` for Standard, `3` for Advanced, `4` for Expert.
4. View the menu at the new object level. Additional configuration options for the Directory Server components are displayed.

Using dsconfig Interactive: Viewing Administrative Alerts

The `dsconfig` tool and the Administrative Console provide a useful feature that displays notifications for certain operations that require further administrator action to complete the process. If you change a certain backend configuration property, the admin action will appear in two places during a `dsconfig` interactive session: when configuring the property and before you apply the change. For example, if you change the `db-directory` property on the userRoot backend (that is, specify the path to the filesystem path that holds the Oracle Berkeley DB Java Edition backend files), you will see an admin action reminder during one of the steps (shown below).

The admin action alert will also appear as a final confirmation step. The alert allows you to continue and apply the change or back out of the configuration if the resulting action cannot be conducted at the present time. For example, after you type "`f`" to apply the `db-directory` property change, the admin alert message appears:

```
Enter choice [b]: f
One or more configuration property changes require administrative action or
confirmation/notification.

Those properties include:

* db-directory: Modification requires that the Directory Server be stopped,
  the database directory manually relocated, and then the Directory Server
  restarted. While the Directory Server is stopped, the directory and files
  pertaining to this backend in the old database directory must be manually
  moved or copied to the new location.

Continue? Choose 'no' to return to the previous step (yes / no) [yes]:
```

Currently, only a small set of properties display an admin action alert appear in `dsconfig` interactive mode and the Administrative Console. For more information on the properties, see the *PingDirectory Server Configuration Reference*.

Using dsconfig in Non-Interactive Mode

The `dsconfig` non-interactive command-line mode provides a simple way to make arbitrary changes to the Directory Server by invoking it from the command line. To use administrative scripts to automate configuration changes, run the `dsconfig` command in non-interactive mode, which is convenient scripting applications. Note, however, that if you plan to make changes to multiple configuration objects at the same time, then the batch mode might be more appropriate.

You can use the `dsconfig` tool to update a single configuration object using command-line arguments to provide all of the necessary information. The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The --no-prompt argument indicates that you want to use non-interactive mode. The {sub-command} is used to indicate which general action to perform. The {globalArgs} argument provides a set of arguments that specify how to connect and authenticate to the Directory Server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on your setup. For example, using standard LDAP connections, you can invoke the `dsconfig` tool as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

If your system uses SASL GSSAPI (Kerberos), you can invoke `dsconfig` as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The {subcommandArgs} argument contains a set of arguments specific to the particular subcommand that you wish to invoke. To always display the advanced properties, use the --advanced command-line option.

Note: Global arguments can appear anywhere on the command line (including before the subcommand, and after or intermingled with subcommand-specific arguments). The subcommand-specific arguments can appear anywhere after the subcommand.

To Configure the Server Using dsconfig Non-Interactive Mode

- Use the `dsconfig` command in non-interactive mode to change the amount of memory used for caching database contents and to specify common parent DNs that should be compacted in the underlying database.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --set db-cache-percent:40 \
  --add "compact-common-parent-dn:ou=accts,dc=example,dc=com" \
  --add "compact-common-parent-dn:ou=subs,dc=example,dc=com"
```

To View a List of dsconfig Properties

1. Use the `dsconfig` command with the list-properties option to view the list of all `dsconfig` properties. Remember to add the LDAP connection parameters.

```
$ bin/dsconfig list-properties
```

2. Use the `dsconfig` command with the list-properties option and the --complexity <menu level> to view objects at and below the menu object level. You can also add the --includeDescription argument that includes a synopsis and description of each property in the output. Remember to add the LDAP connection parameters.

```
$ bin/dsconfig list-properties --complexity advanced --includeDescription
```

3. If the server is offline, you can run the command with the --offline option. You do not need to enter the LDAP connection parameters.

```
 $ bin/dsconfig list-properties --offline --complexity advanced --
includeDescription
```

You can also view the `<server-root>/docs/config-properties.txt` that contains the property information provided with the server.

Getting the Equivalent dsconfig Non-Interactive Mode Command

While the `dsconfig` non-interactive command-line mode is convenient for scripting and automating processes, obtaining the correct arguments and properties for each configuration change can be quite time consuming.

To facilitate easy and quick configuration, you can use an option to display the equivalent non-interactive command using `dsconfig` interactive mode. The command displays the equivalent `dsconfig` command to recreate the configuration in a scripted configuration or to more quickly enter any pending changes on the command line for another server instance.

Note: There are two other ways to get the equivalent `dsconfig` command. One way is by looking at the `logs/config-audit.log`. It might be more convenient to set the Directory Server up the way you want and then get the `dsconfig` arguments from the log. Another way is by configuring an option using the Administrative Console. The console shows the equivalent `dsconfig` command prior to applying the change.

To Get the Equivalent dsconfig Non-Interactive Mode Command

1. Using `dsconfig` in interactive mode, make changes to a configuration but do not apply the changes (that is, do not enter "f").
2. Enter `d` to view the equivalent non-interactive command.
3. View the equivalent command (seen below), and then press **Enter** to continue. For example, based on an example in the previous section, changes made to the `db-cache-percent` returns the following:

   ```
   Command line to apply pending changes to this Local DB Backend:
   dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:40
   ```

   The command does not contain the LDAP connection parameters required for the tool to connect to the host since it is presumed that the command would be used to connect to a different remote host.

Using dsconfig Batch Mode

The PingDirectory Server provides a `dsconfig` batching mechanism that reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The --no-prompt option is required with `dsconfig` in batch mode.

If a `dsconfig` command has a missing or incorrect argument, the command will fail and abort the batch process without applying any changes to the Directory Server. The `dsconfig` command supports a `--batch-continue-on-error` option which instructs `dsconfig` to apply all changes and skip any errors.

You can view the `logs/config-audit.log` file to review the configuration changes made to the Directory Server and use them in the batch file. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The Directory Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to PingDirectory Server machines.

To Configure the Directory Server in dsconfig Batch Mode

1. Create a text file that lists each `dsconfig` command with the complete set of properties that you want to apply to the Directory Server. The items in this file should be in the same format as those accepted by the `dsconfig` command. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

   ```
   # This dsconfig operation creates the exAccountNumber global attribute
    index.
   dsconfig create-global-attribute-index
   --processor-name ou_people_dc_example_dc_com-eb-req-processor
   --index-name exAccountNumber --set prime-index:true

   # Here we create the entry-count placement algorithm with the
   # default behavior of adding entries to the smallest backend
   # dataset first.

   dsconfig create-placement-algorithm
   --processor-name ou_people_dc_example_dc_com-eb-req-processor
   --algorithm-name example_com_entry_count
   --type entry-counter
   --set enabled:true
   --set "poll-interval:1 m"

   # Note that once the entry-count placement algorithm is created
   ```

```
# and enabled, we can delete the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then delete the
# original round-robin algorithm created during the setup
# procedure.

dsconfig delete-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin
```

2. Use `dsconfig` with the --batch-file option to read and execute the commands.

About Recurring Tasks and Task Chains

For regular maintenance items that need to be done on the PingDirectory Server, recurring tasks and task chains can be created with the `dsconfig create-recurring-task` command. These tasks can be created to perform regular backups, LDIF exports, or other static operations. Because this process is owned by the server, tasks do not require special priviledges or credentials, and they can be run when the server is offline. Tasks are created and then added to a recurring task chain for scheduling. The task chain insures that that invocations of a task or set of tasks run in a specified order and cannot overlap.

A recurring task includes:

- The task-specific object classes to include in the task entry.
- The task-specific attributes to include in the task entry, if any.
- Whether to alert on task start, success, and/or failure.
- Any addresses to email on task start, success, and/or failure.
- Whether to cancel an instance of the task if it is dependent upon another task, and that task does not complete successfully.

Once a task is created, one or more tasks can be added to and scheduled with a task chain with the `dsconfig create-recurring-task-chain` command. A recurring task chain includes:

- An ordered list of the tasks to invoke.
- The months, days, times, and time zones in which the task can be scheduled to start.
- The behavior to exhibit if any of the tasks are interrupted by a server shutdown.
- The behavior to exhibit if the server is offline when the start time occurs.

To Create a Recurring Task and Task Chain

Use `dsconfig` to create one or more tasks and then add them to a task chain for scheduling.

1. Create a task. The following creates a backup task.

```
$ bin/dsconfig create-recurring-task \
  --task-name backup-1 \
  --type backup \
  --set 'email-on-failure:admin2@company.com' \
  --set 'email-on-failure:admin@company.com' \
  --set compress:true \
  --set encrypt:true \
  --set "retain-previous-full-backup-age:4 w" \
  --set retain-previous-full-backup-count:10
```

2. Create a task chain to schedule and run recurring tasks.

```
$ bin/dsconfig create-recurring-task-chain \
  --chain-name "backup chain" \
  --set recurring-task:backup-1 \
  --set scheduled-date-selection-type:selected-days-of-the-month \
  --set scheduled-day-of-the-month:last-day-of-the-month \
  --set scheduled-time-of-day:02:00
```

Topology Configuration

Topology configuration enables grouping servers and mirroring configuration changes automatically. It uses a master/slave architecture for mirroring shared data across the topology. All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group. Servers can be added to an existing topology at installation.

Note: To remove a server from the topology, it must be uninstalled with the uninstall tool.

Topology Master Requirements and Selection

A topology master server receives any configuration change from other servers in the topology, verifies the change, then makes the change available to all connected servers. The master always sends a digest of its subtree contents on each update. If the node has a different digest than the master, it knows it's not synchronized. The servers will pull the entire subtree from the master if they detect that they are not synchronized. A server may detect it is not synchronized with the master under the following conditions:

- At the end of its periodic polling interval, if a server's subtree digest differs from that of its master, then it knows it's not synchronized.
- If one or more servers have been added to or removed from the topology, the servers will not be synchronized.

The master of the topology is selected by prioritizing servers by minimum supported product version, most available, newest server version, earliest start time, and startup UUID (a smaller UUID is preferred).

After determining a master, the topology data is reviewed from all available servers (every five seconds by default) to determine if any new information makes a server better suited to being the master. If a new server can be the master, it will communicate that to the other servers, if no other server has advertised that it should be the master. This ensures that all servers accept the same master at approximately the same time (within a few milliseconds of each other). If there is no better master, the initial master maintains the role.

After the best master has been selected for the given interval, the following conditions are confirmed:

- A majority of servers is reachable from that master. (The master server itself is considered while determining this majority.)
- There is only a single master in the entire topology.

If either of these conditions is not met, the topology is without a master and the peer polling frequency is reduced to 100 milliseconds to find a new master as quickly as possible. If there is no master in the topology for more than one minute, a `mirrored-subtree-manager-no-master- found` alarm is raised. If one of the servers in the topology is forced as master with the `force-as-master-for-mirrored-data` option in the Global Configuration object, a `mirrored-subtree-manager-forced-as-master-warning` warning alarm is raised. If multiple servers have been forced as masters, then a `mirrored-subtree-manager-forced-as-master-error` critical alarm will be raised.

Topology Components

When a server is installed, it can be added to an existing topology, which will clone the server's configuration. Topology settings are designed to operate without additional configuration. If required, some settings can be adjusted to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the Global Configuration and in the Config File Handler Backend. Though they are topology settings, they are unique to each server and are not mirrored. Settings must be kept the same on all servers.

The Global Configuration object contains a single topology setting, force-as-master-formirrored- data. This should be set to true on only one of the servers in the topology, and is used only if a situation occurs where the topology cannot determine a master because a majority of servers is not available. A server with this setting enabled will be assigned the role of master, if no suitable master can be determined.

The Config File Handler Backend defines three topology (`mirrored-subtree`) settings:

- `mirrored-subtree-peer-polling-interval` – Specifies the frequency at which the server polls its topology peers to determine if there are any changes that may warrant a new master selection. A lower value will ensure a faster failover, but it will also cause more traffic among the peers. The default value is five seconds. If no suitable master is found, the polling frequency is adjusted to 100 milliseconds until a new master is selected.
- `mirrored-subtree-entry-update-timeout` – Specifies the maximum length of time to wait for an update operation (add, delete, modify or modify-dn) on an entry to be applied by the master on all of the servers in the topology. The default is 10 seconds. In reality, updates can take up to twice as much time as this timeout value if master selection is in progress at the time the update operation was received.
- `mirrored-subtree-search-timeout` – Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

Topology meta-data is stored under the `cn=topology,cn=config` subtree and cluster data is stored under the `cn=cluster,cn=config` subtree. The only setting that can be changed is the cluster name.

Monitor Data for the Topology

Each server has a monitor that exposes that server's view of the topology in its monitor backend, so that peer servers can periodically read this information to determine if there are changes in the topology. Topology data includes the following:

- The server ID of the current master, if the master is not known.
- The instance name of the current master, or if a master is not set, a description stating why a master is not set.
- A flag indicating if this server thinks that it should be the master.
- A flag indicating if this server is the current master.
- A flag indicating if this server was forced as master.
- The total number of configured peers in the topology group.
- The peers connected to this server.
- The current availability of this server.
- A flag indicating whether or not this server is not synchronized with its master, or another node in the topology if the master is unknown.
- The amount of time in milliseconds where multiple masters were detected by this server.
- The amount of time in milliseconds where no suitable server is found to act as master.
- A SHA-256 digest encoded as a base-64 string for the current subtree contents.

The following metrics are included if this server has processed any operations as master:

- The number of operations processed by this server as master.
- The number of operations processed by this server as master that were successful.
- The number of operations processed by this server as master that failed to validate.
- The number of operations processed by this server as master that failed to apply.
- The average amount of time taken (in milliseconds) by this server to process operations as the master.
- The maximum amount of time taken (in milliseconds) by this server to process an operation as the master.

Updating the Server Instance Listener Certificate

To change the SSL certificate for the server, update the keystore and truststore files with the new certificate. The certificate file must have the new certificate in PEM-encoded format, such as:

```
-----BEGIN CERTIFICATE-----

 MIIDKTCCAhGgAwIBAgIEacgGrDANBgkqhkiG9w0BAQsFADBFMR4wHAYDVQQKExVVbmJvdW5kSUQgQ2VydGlmaWNh
 GUxIzAhBgNVBAMTGnZtLW1lZGl1bS03My51bmJvdW5kaWQubGFiMB4XDTE1MTAxMjE1MzU0OFoXDTM1MTAwNzE1

 U0OFowRTEeMBwGA1UEChMVVW5ib3VuZElEIENlcnRpZmljYXRlMSMwIQYDVQQDExp2bS1tZWRpdW0tNzMudW5ib

 uZGlkLmxhYjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKN4tAN3o9Yw6Cr9hivwVDxJqF6+aEi9Ir
            GFYLSrggRNXsiAOfWkSMWdIC5vyF5OJ9DlIgvHL4OuqP/
YNEGzKDkgr6MwtUeVSK14+dCixygJGC0nY7k+f0WSCjt
            IHzrmc4WWdrZXmgb
+qv9LupS30JG0FXtcbGkYpjaKXIEqMg4ekz3B5cAvE0SQUFyXEdN4rWOn96nVFkb2CstbiPzA

 gne2tu7paJ6SGFOW0UF7v018XY1m2WHBIoD0WC8nOVLTG9zFUavaOxtlt1TlhClkI4HRMNg8n2EtSTdQRizKuw9
            TXJBb6Kfvnp/
nI73VHRyt47wUVueehEDfLtDP8pMCAwEAAaMhMB8wHQYDVR0OBBYEFMrwjWxl2K+yd9+Y65oKn0g5
            jITgMA0GCSqGSIb3DQEBCwUAA4IBAQBpsBYodblUGew+HewqtO2i8Wt+vAbt31zM5/
kRvo6/+iPEASTvZdCzIBcgl

 etxKGKeCQ0GPeHr42+erakiwmGDlUTYrU3LU5pTGTDLuR2IllTT5xlEhCWJGWipW4q3Pl3cX/9m2ffY/
JLYDfTJao

 JvnXrh7Sg719skkHjWZQgOHXlkPLx5TxFGhAovE1D4qLVRWGohdpWDrIgFh0DVfoyAn1Ws9ICCXdRayajFI4Lc6
            m6SA5+25Y9nno8BhVPf4q5OW6+UDc8MsLbBsxpwvR6RJ5cv3ypfOriTehJsG
+9ZDo7YeqVsTVGwAlW3PiSd9bYP/8
            yu9Cy+0MfcWcSeAE
            -----END CERTIFICATE-----
```

If clients that already have a secure connection established with this server need to be maintained, information about both certificates can reside in the same file (each with their own begin and end headers and footers).

After the keystore and truststore files are updated, run the following dsconfig command to update the server's certificate in the topology registry:

```
$ bin/dsconfig set-server-instance-listener-prop \
  --instance-name server-instance-name \
  --listener-name ldap-listener-mirrored-config \
  --set listener-certificate path-to-new-certificate-file
```

The `listener-certificate` in the topology registry is like a trust store. The public certificates that it has are automatically trusted by the local server. When the local server attempts a secure LDAP connection to a peer, and the peer presents it with its certificate, the local server will check the `listener-certificate` property for that server in the topology registry. If the property contains the peer server's certificate, the local server will trust the peer.

Remove the Self-signed Certificate

The server is installed with a self-signed certificate and key (`ads-certificate`), which are used for internal purposes such as replication authentication, inter-server authentication in the topology registry, reversible password encryption, and encrypted backup or LDIF export. The ads-certificate lives in the keystore file called `ads-truststore` under the server's `/config` directory. If your deployment requires removing the self-signed certificate, it can be replaced.

The certificate is stored in the topology registry, which enables replacing it on one server and having it mirrored to all other servers in the topology. Any change is automatically mirrored on other servers in the topology. It is stored in human-readable PEM-encoded format and can be updated with dsconfig. The following general steps are required to replace the self-signed certificate:

1. Prepare a new keystore with the replacement key-pair.
2. Update the server configuration to use the new certificate by adding it to the server's list of certificates in the topology registry so that it is trusted by other servers.
3. Update the server's `ads-truststore` file to use the new key-pair.
4. Retire the old certificate by removing it from the topology registry.

Note: Replacing the entire key-pair instead of just the certificate associated with the original private key can make existing backups and LDIF exports invalid. This should be performed immediately after setup or before the key-pair is used. After the first time, only the certificate associated with the private key should have to be changed, for example, to extend its validity period or replace it with a certificate signed by a different CA.

Prepare a New Keystore with the Replacement Key-pair

The self-signed certificate can be replaced with an existing key-pair, or the certificate associated with the original key-pair can be used.

To Use an Existing Key-pair

If a private key and certificate(s) in PEM-encoded format already exist, both the original private key and self-signed certificate can be replaced in ads-truststore with the `manage-certificates` tool.

- The following command imports the keystore file, ads-truststore.new.

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file existing.key \
  --certificate-file existing.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

The certificates listed using the `--certificate-file` options must be ordered so that each subsequent certificate is the issuer for the previous one. So the server certificate comes first, the intermediate certificates next (if any), and the root CA certificate last.

To Use the Certificate Associated with the Original Key-pair

The certificate associated with the original server-generated private key can be replaced with the following commands:

1. Create a CSR for the `ads-certificate`:

```
$ bin/manage-certificates generate-certificate-signing-request \
  --keystore ads-truststore \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --use-existing-key-pair \
  --subject-dn "CN=ldap.example.com,O=Example Corporation,C=US" \
  --output-file ads.csr
```

2. Submit `ads.csr` to a CA for signing.
3. Export the server's private key into `ads.key`:

```
$ bin/manage-certificates export-private-key \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --output-file ads.key
```

4. Import the certificates obtained from the CA (the CA-signed server certificate, any intermediate certificates, and root CA certificate) into `ads-truststore.new`:

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file ads.key \
  --certificate-file new-ads.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

To Update the Server Configuration to Use the New Certificate

To update the server to use the desired key-pair, the `inter-server-certificate` property for the server instance must first be updated in the topology registry. The old and the new certificates may appear within their own begin and end headers in the `inter-server-certificate` property to support transitioning from the old certificate to the new one.

1. Export the server's old ads-certificate into `old-ads.crt`::

```
$ bin/manage-certificates export-certificate \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --export-certificate-chain \
  --output-file old-ads.crt
```

2. Concatenate the old, new certificate, and issuer certificates into one file. On Windows, an editor like notepad can be used. On Unix platforms, use the following command:

```
$ cat old-ads.crt new-ads.crt intermediate.crt root-ca.crt > chain.crt
```

3. Update the `inter-server-certificate` property for the server instance in the topology registry using `dsconfig`:

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name instance-name \
  --set "inter-server-certificate<chain.crt"
```

To Update the ads-truststore File to Use the New Key-pair

The server will still use the old `ads-certificate`. When the new `ads-certificate` needs to go into effect, the old `ads-truststore` file must be replaced with `ads-truststore.new` in the server's `config` directory.

- Move the file.

```
$ mv ads-truststore.new ads-truststore
```

To Retire the Old Certificate

The old certificate is retired by removing it from the topology registry when it has expired. All existing encrypted backups and LDIF exports are not affected because the public key in the old and new server certificates are the same, and the private key will be able to decrypt them.

- Perform the following commands:

```
$ cat new-ads.crt intermediate.crt root-ca.crt<chain.crt
```

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name instance-name \
  --set "inter-server-certificate<chain.crt"
```

Using the Configuration API

PingData servers provide a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API is consistent with the System for Cross-domain Identity Management

(SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers should allow the application/json content type.

The server includes a servlet extension that provides read and write access to the server's configuration over HTTP. The extension is enabled by default for new installations, and can be enabled for existing deployments by simply adding the extension to one of the server's HTTP Connection Handlers, as follows:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add http-servlet-extension:Configuration
```

The API is made available on the HTTPS Connection handler's host:port in the `/config` context. Due to the potentially sensitive nature of the server's configuration, the HTTPS Connection Handler should be used for hosting the Configuration extension.

Authentication and Authorization with the Configuration API

Clients must use HTTP Basic authentication to authenticate to the Configuration API. If the username value is not a DN, then it will be resolved to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper, or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. For example:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Configuration \
  --set "identity-mapper:Alternative Identity Mapper”
```

To access configuration information, users must have the appropriate privileges:

- To access the `cn=config` backend, users must have the `bypass-acl` privilege or be allowed access to the configuration using an ACI.
- To read configuration information, users must have the `config-read` privilege.
- To update the configuration, users must have the `config-write` privilege.

Relationship Between the Configuration API and the dsconfig Tool

The Configuration API is designed to mirror the `dsconfig` tool, using the same names for properties and object types. Property names are presented as hyphen case in `dsconfig` and as camel-case attributes in the API. In API requests that specify property names, case is not important. Therefore, `baseDN` is the same as `baseDn`. Object types are represented in hyphen case. API paths mirror what is in `dsconfig`. For example, the `dsconfig list-connection-handlers` command is analogous to the API's `/config/connection-handlers` path. Object types that appear in the schema URNs adhere to a `type:subtype` syntax. For example, a Local DB Backend's schema URN is `urn:unboundid:schemas:configuration:2.0:backend:local-db`. Like the `dsconfig` tool, all configuration updates made through the API are recorded in `logs/config-audit.log`.

The API includes the filter, sort, and pagination query parameters described by the SCIM specification. Specific attributes may be requested using the attributes query parameter, whose value must be a comma-delimited list of properties to be returned, for example `attributes=baseDN,description`. Likewise, attributes may be excluded from responses by specifying the `excludedAttributes` parameter.

Operations supported by the API are those typically found in REST APIs:

HTTP Method	Description	Related dsconfig Example
GET	Lists the properties of an object when used with a path representing an object, such as `/config/global-configuration` or `/config/backends/userRoot`. Can also list objects when used with a path representing a parent relation, such as `/config/backends`.	`get-backend-prop`, `list-backends`, `get-global-configuration-prop`

HTTP Method	Description	Related dsconfig Example
POST	Creates a new instance of an object when used with a relation parent path, such as `/config/backends`.	`create-backend`
PUT	Replaces the existing properties of an object. A PUT operation is similar to a PATCH operation, except that the PATCH identifies the difference between an existing target object and a supplied source object. Only those properties in the source object are modified in the target object. The target object is specified using a path, such as `/config/backends/userRoot`.	`set-backend-prop`, `set-global-configuration-prop`
PATCH	Updates the properties of an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`set-backend-prop`, `set-global-configuration-prop`
DELETE	Deletes an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`delete-backend`

The OPTIONS method can also be used to determine the operations permitted for a particular path.

Object names, such as `userRoot` in the Description column, must be URL-encoded for use in the path segment of a URL. For example, `%20` must be used in place of spaces, and `%25` is used in place of the percent (%) character. The URL for accessing the HTTP Connection Handler object is:

```
/config/connection-handlers/http%20connection%20handler
```

GET Example

The following is a sample GET request for information about the `userRoot` backend:

```
GET /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
```

The response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
   "id": "userRoot",
   "meta": {
     "resourceType": "Local DB Backend",
     "location": "http://localhost:5033/config/backends/userRoot"
   },
   "backendID": "userRoot2",
   "backgroundPrime": "false",
   "backupFilePermissions": "700",
   "baseDN": [
     "dc=example2,dc=com"
   ],
   "checkpointOnCloseCount": "2",
   "cleanerThreadWaitTime": "120000",
   "compressEntries": "false",
   "continuePrimeAfterCacheFull": "false",
   "dbBackgroundSyncInterval": "1 s",
   "dbCachePercent": "10",
   "dbCacheSize": "0 b",
   "dbCheckpointerBytesInterval": "20 mb",
   "dbCheckpointerHighPriority": "false",
   "dbCheckpointerWakeupInterval": "1 m",
   "dbCleanOnExplicitGC": "false",
   "dbCleanerMinUtilization": "75",
   "dbCompactKeyPrefixes": "true",
   "dbDirectory": "db",
   "dbDirectoryPermissions": "700",
   "dbEvictorCriticalPercentage": "0",
   "dbEvictorLruOnly": "false",
   "dbEvictorNodesPerScan": "10",
   "dbFileCacheSize": "1000",
   "dbImportCachePercent": "60",
   "dbLogFileMax": "50 mb",
   "dbLoggingFileHandlerOn": "true",
   "dbLoggingLevel": "CONFIG",
   "dbNumCleanerThreads": "0",
   "dbNumLockTables": "0",
   "dbRunCleaner": "true",
   "dbTxnNoSync": "false",
   "dbTxnWriteNoSync": "true",
   "dbUseThreadLocalHandles": "true",
   "deadlockRetryLimit": "10",
   "defaultCacheMode": "cache-keys-and-values",
   "defaultTxnMaxLockTimeout": "10 s",
   "defaultTxnMinLockTimeout": "10 s",
   "enabled": "false",
   "explodedIndexEntryThreshold": "4000",
   "exportThreadCount": "0",
   "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
   "externalTxnDefaultMaxLockTimeout": "100 ms",
   "externalTxnDefaultMinLockTimeout": "100 ms",
   "externalTxnDefaultRetryAttempts": "2",
   "hashEntries": "false",
   "id2childrenIndexEntryLimit": "66",
   "importTempDirectory": "import-tmp",
   "importThreadCount": "16",
   "indexEntryLimit": "4000",
   "isPrivateBackend": "false",
   "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
   "jeProperty": [
     "je.cleaner.adjustUtilization=false",
     "je.nodeMaxEntries=32"
   ],
   "numRecentChanges": "50000",
   "offlineProcessDatabaseOpenTimeout": "1 h",
   "primeAllIndexes": "true",
   "primeMethod": [
     "none"
   ],
   "primeThreadCount": "2",
   "primeTimeLimit": "0 ms",
   "processFiltersWithUndefinedAttributeTypes": "false",
   "returnUnavailableForUntrustedIndex": "true",
   "returnUnavailableWhenDisabled": "true",
   "setDegradedAlertForUntrustedIndex": "true",
   "setDegradedAlertWhenDisabled": "true",
   "subtreeDeleteBatchSize": "5000",
   "subtreeDeleteSizeLimit": "5000",
   "uncachedId2entryCacheMode": "cache-keys-only",
   "writabilityMode": "enabled"
}
```

GET List Example

The following is a sample GET request for all local backends:

```
GET /config/backends/
Host: example.com:5033
Accept: application/scim+json
```

The response (which has been shortened):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=topology,cn=config"
      ],
      "enabled": "true",
      "isPrivateBackend": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.LDIFBackend",
      "ldifFile": "config/admin-backend.ldif",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "false",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
      ],
      "id": "ads-truststore",
      "meta": {
        "resourceType": "Trust Store Backend",
        "location": "http://localhost:5033/config/backends/ads-
truststore"
      },
      "backendID": "ads-truststore",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=ads-truststore"
      ],
      "enabled": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.TrustStoreBackend",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "true",
      "trustStoreFile": "config/server.keystore",
      "trustStorePin": "********",
      "trustStoreType": "JKS",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:alarm"
      ],
      "id": "alarms",
      "meta": {
        "resourceType": "Alarm Backend",
        "location": "http://localhost:5033/config/backends/alarms"
      },
  ...
```

PATCH Example

Configuration can be modified using the HTTP PATCH method. The PATCH request body is a JSON object formatted according to the SCIM patch request. The Configuration API, supports a subset of possible values for the path attribute, used to indicate the configuration attribute to modify.

The configuration object's attributes can be modified in the following ways. These operations are analogous to the `dsconfig modify-[object]` options.

- An operation to set the single-valued `description` attribute to a new value:

  ```
  {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop
    --backend-name userRoot \
    --set "description:A new backend"
  ```

- An operation to add a new value to the multi-valued `jeProperty` attribute:

  ```
  {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }
  ```

  is analogous to:

  ```
  $ dsconfig  set-backend-prop --backend-name userRoot \
    --add je-property:je.env.backgroundReadLimit=0
  ```

- An operation to remove a value from a multi-valued property. In this case, path specifies a SCIM filter identifying the value to remove:

  ```
  {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --remove je-property:je.cleaner.adjustUtilization=false
  ```

- A second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify, and a SCIM filter whose attribute is value:

  ```
  {
  ```

```
  "op" : "remove",
  "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --remove je-property:je.nodeMaxEntries=32
```

- An option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value:

```
{
  "op" : "remove",
  "path" : "id2childrenIndexEntryLimit"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --reset id2childrenIndexEntryLimit
```

The following is the full example request. The API responds with the entire modified configuration object, which may include a SCIM extension attribute `urn:unboundid:schemas:configuration:messages` containing additional instructions:

```
PATCH /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json

{
  "schemas" : [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  } ]
}
```

Example response:

```
{
 "schemas": [
   "urn:unboundid:schemas:configuration:2.0:backend:local-db"
 ],
 "id": "userRoot2",
 "meta": {
   "resourceType": "Local DB Backend",
   "location": "http://example.com:5033/config/backends/userRoot2"
 },
 "backendID": "userRoot2",
 "backgroundPrime": "false",
 "backupFilePermissions": "700",
 "baseDN": [
   "dc=example2,dc=com"
 ],
 "checkpointOnCloseCount": "2",
 "cleanerThreadWaitTime": "120000",
 "compressEntries": "false",
 "continuePrimeAfterCacheFull": "false",
 "dbBackgroundSyncInterval": "1 s",
 "dbCachePercent": "10",
 "dbCacheSize": "0 b",
 "dbCheckpointerBytesInterval": "20 mb",
 "dbCheckpointerHighPriority": "false",
 "dbCheckpointerWakeupInterval": "1 m",
 "dbCleanOnExplicitGC": "false",
 "dbCleanerMinUtilization": "75",
 "dbCompactKeyPrefixes": "true",
 "dbDirectory": "db",
 "dbDirectoryPermissions": "700",
 "dbEvictorCriticalPercentage": "0",
 "dbEvictorLruOnly": "false",
 "dbEvictorNodesPerScan": "10",
 "dbFileCacheSize": "1000",
 "dbImportCachePercent": "60",
 "dbLogFileMax": "50 mb",
 "dbLoggingFileHandlerOn": "true",
 "dbLoggingLevel": "CONFIG",
 "dbNumCleanerThreads": "0",
 "dbNumLockTables": "0",
 "dbRunCleaner": "true",
 "dbTxnNoSync": "false",
 "dbTxnWriteNoSync": "true",
 "dbUseThreadLocalHandles": "true",
 "deadlockRetryLimit": "10",
 "defaultCacheMode": "cache-keys-and-values",
 "defaultTxnMaxLockTimeout": "10 s",
 "defaultTxnMinLockTimeout": "10 s",
 "description": "123",  "enabled": "false",
 "explodedIndexEntryThreshold": "4000",
 "exportThreadCount": "0",
 "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
 "externalTxnDefaultMaxLockTimeout": "100 ms",
 "externalTxnDefaultMinLockTimeout": "100 ms",
 "externalTxnDefaultRetryAttempts": "2",
 "hashEntries": "false",
 "importTempDirectory": "import-tmp",
 "importThreadCount": "16",
 "indexEntryLimit": "4000",
 "isPrivateBackend": "false",
 "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
 "jeProperty": [  "\"je.env.backgroundReadLimit=0\""
 ],
 "numRecentChanges": "50000",
 "offlineProcessDatabaseOpenTimeout": "1 h",
 "primeAllIndexes": "true",
 "primeMethod": [
   "none"
 ],
 "primeThreadCount": "2",
 "primeTimeLimit": "0 ms",
 "processFiltersWithUndefinedAttributeTypes": "false",
 "returnUnavailableForUntrustedIndex": "true",
 "returnUnavailableWhenDisabled": "true",
 "setDegradedAlertForUntrustedIndex": "true",
 "setDegradedAlertWhenDisabled": "true",
 "subtreeDeleteBatchSize": "5000",
 "subtreeDeleteSizeLimit": "5000",
 "uncachedId2entryCacheMode": "cache-keys-only",
 "writabilityMode": "enabled",
 "urn:unboundid:schemas:configuration:messages:2.0": {
   "requiredActions": [
     {
       "property": "jeProperty",
       "type": "componentRestart",
       "synopsis": "In order for this modification to take effect,
          the component must be restarted, either by disabling and
          re-enabling it, or by restarting the server"
     },
     {
       "property": "id2childrenIndexEntryLimit",
       "type": "other",
       "synopsis": "If this limit is increased, then the contents
          of the backend must be exported to LDIF and re-imported to
          allow the new limit to be used for any id2children keys
          that had already hit the previous limit."
     }
   ]
 }
}
```

Configuration API Paths

The Configuration API is available under the /config path. A full listing of supported sub-paths is available by accessing the base `/config/ResourceTypes` endpoint:

```
GET /config/ResourceTypes
Host: example.com:5033
Accept: application/scim+json
```

Sample response (abbreviated):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "dsee-compat-access-control-handler",
      "name": "DSEE Compat Access Control Handler",
      "description": "The DSEE Compat Access Control
              Handler provides an implementation that uses syntax
              compatible with the Sun Java System Directory Server
              Enterprise Edition access control handler.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/dsee-compat-
access-control-handler"
      }
    },
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "access-control-handler",
      "name": "Access Control Handler",
      "description": "Access Control Handlers manage the
              application-wide access control. The server's access
              control handler is defined through an extensible
              interface, so that alternate implementations can be created.
              Only one access control handler may be active in the server
                       at any given time.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/access-
control-handler"
      }
    },
    {
 ...
```

The response's `endpoint` elements enumerate all available sub-paths. The path `/config/access-control-handler` in the example can be used to get a list of existing access control handlers, and create new ones. A path containing an object name such as `/config/backends/{backendName}`, where `{backendName}` corresponds to an existing backend (such as `userRoot`) can be used to obtain an object's properties, update the properties, or delete the object.

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted nor created. These paths can be differentiated from others by their singular, rather than plural, relation name (for example `global-configuration`).

Sorting and Filtering Objects

The Configuration API supports SCIM parameters for filter, sorting, and pagination. Search operations can specify a SCIM filter used to narrow the number of elements returned. See the SCIM specification for the full set of operations for SCIM filters. Clients can also specify sort parameters, or paging parameters. Include or exclude attributes can be specified in both get and list operations.

GET Parameter	Description
filter	Values can be simple SCIM filters such as `id eq "userRoot"` or compound filters like `meta.resourceType eq "Local DB Backend"` and `baseDn co "dc=exmple,dc=com"`.
sortBy	Specifies a property value by which to sort.
sortOrder	Specifies either `ascending` or `descending` alphabetical order.
startIndex	1-based index of the first result to return.
count	Indicates the number of results per page.

Updating Properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH. With PUT, the server computes the differences between the object in the request with the current version in the server, and performs modifications where necessary. The server will never remove attributes that are not specified in the request. The API responds with the entire modified object.

Request:

```
PUT /config/backends/userRoot
```

```
Host: example.com:5033
Accept: application/scim+json
{
  "description" : "A new description."
}
```

Response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode":
  "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "abc",
  "enabled": "true",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior":
  "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "true",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "numRecentChanges": "50000",  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative Actions

Updating a property may require an administrative action before the change can take effect. If so, the server will return `200 Success`, and any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

For example, changing the `jeProperty` of a backend will result in the following:

```
"urn:unboundid:schemas:configuration:messages:2.0": {
    "required-actions": [
      {
       "property": "baseContextPath",
       "type": ""componentRestart",
       "synopsis": "In order for this modification to take effect, the
 component
                 must be restarted, either by disabling and re-enabling it,
 or
                 by restarting the server"
      },
      {
        "property": {
        "type": "other",
        "synopsis": "If this limit is increased, then the
                 contents of the backend must be exported to LDIF
                 and re-imported to allow the new limit to be used
                 for any id2children keys that had already hit the
                 previous limit."
      }
    ]
  }
```

Updating Servers and Server Groups

Servers can be configured as part of a server group, so that configuration changes that are applied to a single server, are then applied to all servers in a group. When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query attribute. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `single-server` or `server-group` can be specified. For example:

```
http://localhost:8082/config/backends/userRoot?applyChangeTo=single-server
```

Configuration API Responses

Clients of the API should examine the HTTP response code in order to determine the success or failure of a request. The following are response codes and their meanings:

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, or object properties, administrative actions.
204 No Content	The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation.	None.
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message.
401 Unauthorized	User authentication is required. Some user agents such as browsers may respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid.	None.
403 Forbidden	The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted.	None.
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message.
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists, or an attempt was made to delete an object that is referred to by another object.	Error summary and optional message.
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None.
500 Server Error	The server encountered an unexpected error. Please report server errors to customer support.	Error summary and optional message.

An application that uses the Configuration API should limit dependencies on particular text appearing in error message content. These messages may change, and their presence may depend on server configuration. Use the HTTP return code and the context of the request to create a client error message. The following is an example encoded error message:

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "status": 404,
  "scimType": null,
  "detail": "The Local DB Index does not exist."
}
```

Configure the Server Using the Administrative Console

The PingDirectory Server provides a Administrative Console for server configuration and monitoring that has the same functionality as that of the `dsconfig` command. When logging on to the Administrative Console, the console does not persistently store any credentials for authenticating to the Directory Server but uses the credentials provided by the user when logging in. When managing multiple directory server instances, the provided credentials must be valid for each instance.

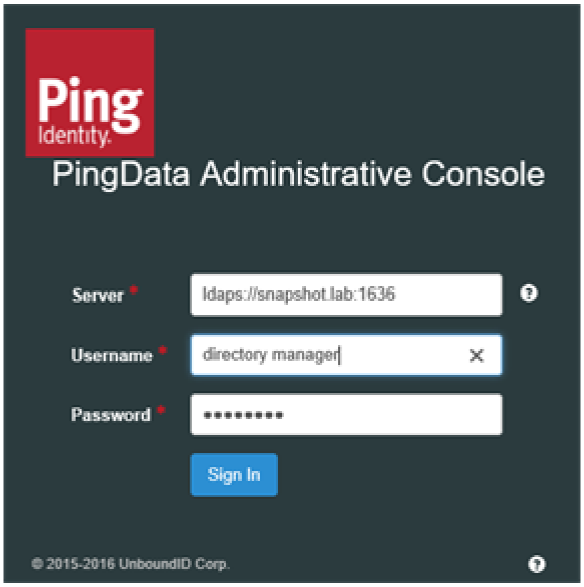
To Log onto the Administrative Console

To log into the console, enter a fully qualified DN (for example, `cn=admin2,cn=Topology Admin Users,cn=Topology,cn=config`). See Configuring a Global Administrator for instructions.

1. Start the Directory Server.

```
$ bin/start-server
```

2. Open a browser to http://server-name:389/console/login.
3. Enter the root user DN and password, then click **Login**.



Server * ldaps://snapshot.lab:1636

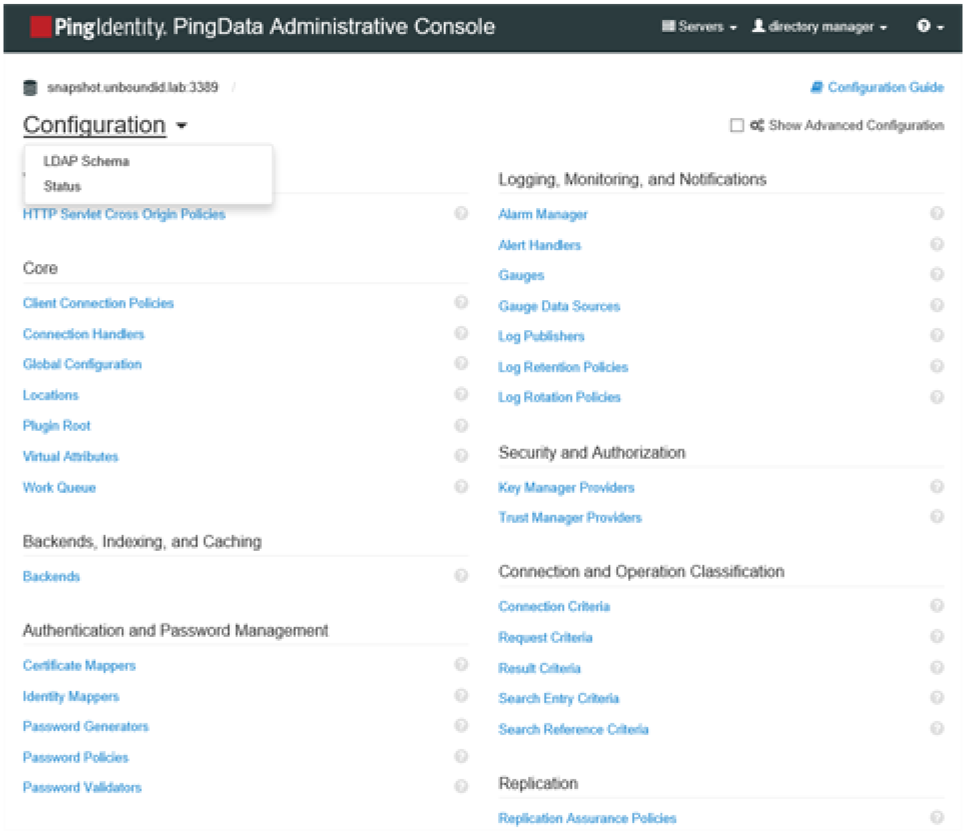
Username * directory manager

Password * •••••••••

The console does not persistently store any credentials for accessing the Directory Server. Instead, it uses the credentials provided by the user when logging into the console. When managing multiple directory server instances, the provided credentials must be valid at each instance.

To Configure the Server Using the Console

1. Log into the Administrative Console. Click **Configuration** to open the **Configuration** menu, and then click **Backends**.



PingIdentity. PingData Administrative Console

Servers ▾ directory manager ▾

snapshot.unboundid.lab:3389 /

Configuration Guide

Configuration ▾

□ Show Advanced Configuration

LDAP Schema

Status

HTTP Servlet Cross Origin Policies

Core

Client Connection Policies

Connection Handlers

Global Configuration

Locations

Plugin Root

Virtual Attributes

Work Queue

Backends, Indexing, and Caching

Backends

Authentication and Password Management

Certificate Mappers

Identity Mappers

Password Generators

Password Policies

Password Validators

Logging, Monitoring, and Notifications

Alarm Manager

Alert Handlers

Gauges

Gauge Data Sources

Log Publishers

Log Retention Policies

Log Rotation Policies

Security and Authorization

Key Manager Providers

Trust Manager Providers

Connection and Operation Classification

Connection Criteria

Request Criteria

Result Criteria

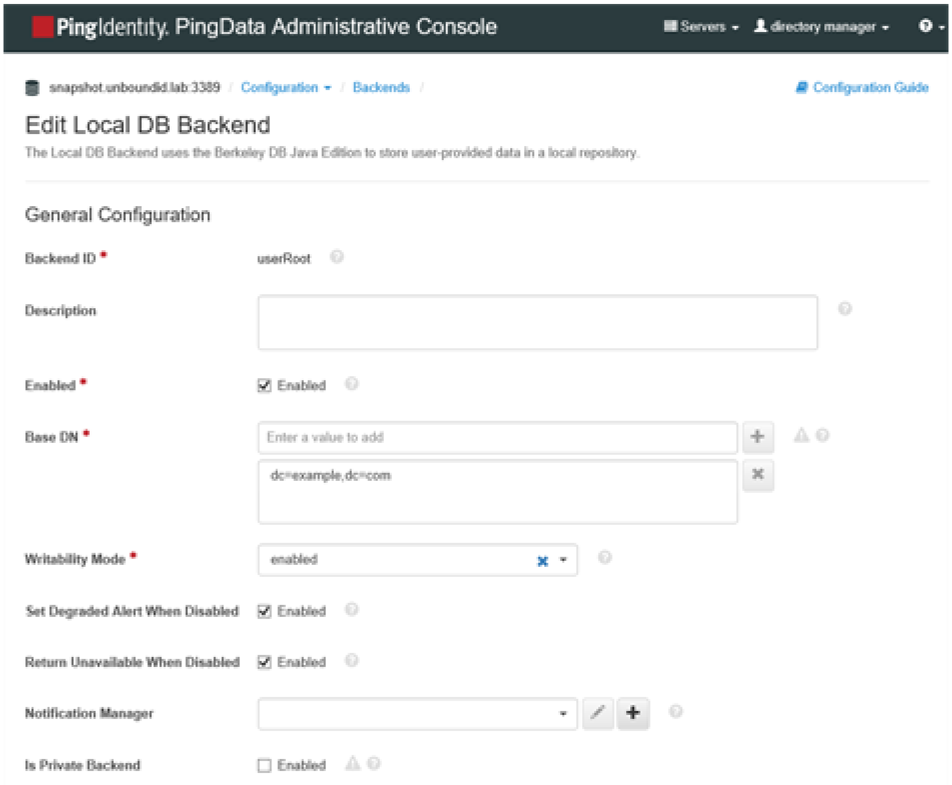
Search Entry Criteria

Search Reference Criteria

Replication

Replication Assurance Policies

2. At the top of the Administrative Console page, click **Show Advanced Properties**. Click **Backends**. For this example, click **userRoot**.



PingIdentity. PingData Administrative Console

Servers ▾ directory manager ▾

snapshot.unboundid.lab:3389 / Configuration ▾ / Backends /

Configuration Guide

Edit Local DB Backend

The Local DB Backend uses the Berkeley DB Java Edition to store user-provided data in a local repository.

General Configuration

Backend ID * userRoot

Description

Enabled * ☑ Enabled

Base DN * Enter a value to add

dc=example,dc=com

Writability Mode * enabled

Set Degraded Alert When Disabled ☑ Enabled

Return Unavailable When Disabled ☑ Enabled

Notification Manager

Is Private Backend □ Enabled

3. Change or enter values in the `userRoot` configuration. For this example, change the `DB Cache Percent` value to 40, and then click **Save**.

Generating a Summary of Configuration Components

The Directory Server provides a `config-diff` tool that generates a summary of the configuration in a local or remote directory server instance. The tool is useful when comparing configuration settings on the directory server instance when troubleshooting issues or when verifying configuration settings on newly-added servers to your network. The tool can interact with the local configuration regardless of whether the server is running or not.

Run the `config-diff` --help option to view other available tool options.

To Generate a Summary of Configuration Components

- Run the `config-diff` tool to generate a summary of the configuration components on the directory server instance. The following command runs a summary on a local online server.

```
$ bin/config-diff
```

- The following example compares the current configuration of the local server to the baseline, pre-installation configuration, ignoring any changes that could be made by the installer, and writes the output to the `configuration-steps.dsconfig` file. This provides a script that can be used to configure a newly installed server identically to the local server:

```
$ bin/config-diff --sourceLocal \
  --sourceBaseline \
  --targetLocal \
  --exclude differs-after-install \
  --outputFile configuration-steps.dsconfig
```

About Root User, Administrator, and Global Administrators

The Directory Server provides three different classes of administrator accounts: root user, administrator, and global administrator. The root user is the LDAP-equivalent of a UNIX super-user account and inherits its privileges from the default root user privilege set (see Default Root Privileges). The root user "account" is an entry that is stored in the server's configuration under the `cn=Root DNs,cn=config` and bypasses access control evaluation, and can be created manually, or with the `dsconfig` tool. This account has full access to the entire set of data in the Directory Information Tree (DIT) as well as full access to the server configuration and its operations. One important difference between other vendors' servers and the Directory Server's implementation is that the root user's rights are granted through a set of privileges. This allows the Directory Server to have multiple root users on its system if desired; however, the normal practice is to set up administrator user entries. Also, by default, the Root User has no resource limits.

The administrator user can have a full set of root user privileges but often has a subset of these privileges to limit the accessible functions that can be performed. The administrators entries typically have limited access to the entire set of data in the directory information tree (DIT), which is controlled by access control instructions. These entries reside in the backend configuration (for example, `uid=admin,dc=example,dc=com`) and are replicated between servers in a replication topology. In some cases, administrator user accounts may be unavailable when the server enters lockdown mode unless the administrator is given the lock-down mode privilege.

A global administrator is primarily responsible for managing configuration server groups. A configuration server group is an administration domain that allows you to synchronize configuration changes to one or all of the servers in the group. For example, you can set up a group when configuring a replication topology, where configuration changes to one server can be applied to all of the servers at one time. Global Administrator entries are stored in the `cn=Topology Admin Users,cn=Topology,cn=config` backend are always mirrored across servers in a replication topology. These users can be assigned privileges like other admin users but are typically used to manage the data under `cn=Topology,cn=config`.

Managing Root Users Accounts

The PingDirectory Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file (for example, under `cn=Root DNs,cn=config`). The root user is the LDAP-equivalent of a UNIX super-user account and inherits its read-write privileges from the default root privilege set. Root users can be created and updated with the `dsconfig` tool. Root user entries are stored in the server’s configuration.The following is a sample command to create a new root user:

```
bin/dsconfig create-root-dn-user --user-name "Joanne Smith" \
  --set last-name:Smith \
  --set first-name:Joanne \
  --set user-id:jsmith \
  --set 'email-address:jsmith@example.com' \
  --set mobile-telephone-number:8889997777 \
  --set home-telephone-number:5556667777 \
  --set work-telephone-number:4445556666
```

To limit full access to all of the Directory Server, create separate administrator accounts with limited privileges so that you can identify the administrator responsible for a particular change. Having separate user accounts for each administrator also makes it possible to enable password policy functionality (such as password expiration, password history, and requiring secure authentication) for each administrator.

Default Root Privileges

The PingDirectory Server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments.

Note: Creating restricted root user accounts requires assigning privileges and necessary access controls for actions on specific data or backends. Access controls are determined by how the directory is configured and the structure of your data. See Chapter 16: Managing Access Controls for more information.

The following set of root privileges are available to each root user DN:

Table 4: Default Root Privileges

Privilege	Description
audit-data-security	Allows the associated user to execute data security auditing tasks.
backend-backup	Allows the user to perform backend backup operations.
backend-restore	Allows the user to perform backend restore operations.
bypass-acl	Allows the user to bypass access control evaluation.
config-read	Allows the user to read the server configuration.
config-write	Allows the user to update the server configuration.
disconnect-client	Allows the user to terminate arbitrary client connections.
ldif-export	Allows the user to perform LDIF export operations.
ldif-import	Allows the user to perform LDIF import operations.
lockdown-mode	Allows the user to request a server lockdown.
manage-topology	Allows the user to modify topology setting.
metrics-read	Allows the user to read server metrics.
modify-acl	Allows the user to modify access control rules.

Privilege	Description
password-reset	Allows the user to reset user passwords but not their own. The user must also have privileges granted by access control to write the user password to the target entry.
permit-get-password-policy-state-issues	Allows the user to access password policy state issues.
privilege-change	Allows the user to change the set of privileges for a specific user, or to change the set of privileges automatically assigned to a root user.
server-restart	Allows the user to request a server restart.
server-shutdown	Allows the user to request a server shutdown.
soft-delete-read	Allows the user access to soft-deleted entries.
stream-values	Allows the user to perform a stream values extended operation that obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
third-party-task	Allows the associated user to invoke tasks created by third-party developers.
unindexed-search	Allows the user to perform an unindexed search in the Oracle Berkeley DB Java Edition backend.
update-schema	Allows the user to update the server schema.
use-admin-session	Allows the associated user to use an administrative session to request that operations be processed using a dedicated pool of worker threads.

The Directory Server provides other privileges that are not assigned to the root user DN by default but can be added using the `ldapmodify` tool (see Modifying Individual Root User Privileges) for more information.

Table 5: Other Available Privileges

Privilege	Description
bypass-pw-policy	Allows the associated user bypass password policy rules and restrictions.
bypass-read-aci	Allows the associated user to bypass access control checks performed by the server for bind, compare, and search operations. Access control evaluation may still be enforced for other types of operations.
jmx-notify	Allows the associated user to subscribe to receive JMX notifications.
jmx-read	Allows the associated user to perform JMX read operations.
jmx-write	Allows the associated user to perform JMX write operations.
permit-externally-processed-authentication	Allows the associated user accept externally processed authentication.
permit-proxied-mschapv2-details	Allows the associated user to permit MS-CHAP V2 handshake protocol.
proxied-auth	Allows the associated user to accept proxied authorization.

Configuring Administrator Accounts

An administrator account is any account in the user backend that is assigned one or more privileges, or given access to read and write operations beyond that of a normal user entry. The privilege mechanism is the same as that used for Root DN accounts and allows individual privileges to be assigned to an administrator entry.

Typically, administrator user entries are controlled by access control evaluation to limit access to the entire set of data in the Directory Information Tree (DIT). Fine-grained read and write access can be granted using the access control definitions available via the `aci` attribute. Administrator entries reside in the backend configuration (for example, `uid=admin,dc=example,dc=com`) and are replicated between servers in a replication topology.

The following examples show how to configure administrator accounts. The first procedure shows how to set up a single, generic `uid=admin,dc=example,dc=com` account with limited privileges. Note that if you generated sample data at install, you can view an example `uid=admin` entry using `ldapsearch`. The second example shows a more realistic example, where the user is part of the Administrators group. Note that both examples are based on a simple DIT. Actual deployment cases depends on your schema.

To Set Up a Single Administrator Account

1. Create an LDIF file with an example Administrator entry.

```
dn: uid=admin,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: Admin
uid: admin
cn: Admin User
sn: User
userPassword: password
```

2. Then add the entry using the `ldapmodify` tool.

```
$ bin/ldapmodify --defaultAdd --filename admin.ldif
```

3. Create another LDIF file to add the access control instruction (ACI) to the root suffix, or base DN to give full access to the new administrator. The ACI grants full access to all user attributes, but not to operational attributes. If you want to grant access to operational attributes as well as user attributes, use `(targetattr = "*||+")` in the access control instruction.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

4. Then add the entry using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename admin.ldif
```

5. Verify the additions using `ldapsearch`. The first command searches for the entry that contains `uid=Admin` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including access control instructions, associated with the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=admin)"

$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope base
 "(objectclass=*)" "+"
```

6. Add specific privileges to the Admin account. In this example, add the `password-reset` privilege to the admin account from the command line. After typing the privileges, press **CTRL-D** to process the modify operation.

```
$ bin/ldapmodify
dn: uid=admin,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset
```

```
Processing MODIFY request for uid=admin,dc=example,dc=com
MODIFY operation successful for DN uid=admin,dc=example,dc=com
```

7. Assign a password policy for the Admin account. For example, create an "Admin Password Policy", then add the password policy to the account.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Admin Password Policy" \
  --set "description:Password policy for administrators" \
  --set password-attribute:userpassword \
  --set "default-password-storage-scheme:Salted 256-bit SHA-2" \
  --set password-change-requires-current-password:true \
  --set force-change-on-reset:true \
  --set "max-password-age:25w 5d" \
  --set grace-login-count:3
  --no-prompt
```

8. Apply the password policy to the account. In this example, the password policy is being added from the command line. The following `ldapmodify` command should be executed with a bind DN that has sufficient rights, such as a Root DN.

```
$ bin/ldapmodify
dn: uid=admin,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Admin Password Policy,cn=Password
 Policies,cn=config
```

To Change the Administrator Password

Root users are governed by the Root Password Policy and by default, their passwords never expire. However, if a root user password must be changed, use the `ldappasswordmodify` tool.

1. Open a text editor and create a text file containing the new password. In this example, name the file `rootuser.txt`.

```
$ echo password > rootuser.txt
```

2. Use `ldappasswordmodify` to change the root user's password.

```
$ bin/ldappasswordmodify --port 1389 --bindDN "cn=Directory Manager"\
  --bindPassword secret --newPasswordFile rootuser.txt
```

3. Remove the text file.

```
$ rm rootuser.txt
```

To Set Up an Administrator Group

The following example shows how to set up a group of administrators that have access rights to the whole Directory Server. The example uses a static group using the `GroupOfUniqueNames` object class.

1. Create an LDIF file with an example Administrator group, and save it as `admin-group.ldif`.

```
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: Groups

dn: cn=Dir Admins,ou=Groups,dc=example,dc=com
objectClass: groupofuniquenames
objectClass: top
uniqueMember: uid=user.0, ou=People, dc=example,dc=com
```

```
uniqueMember: uid=user.1, ou=People, dc=example,dc=com
cn: Dir Admins
ou: Groups
```

2. Then, add the entries using the `ldapmodify` tool.

   ```
   $ bin/ldapmodify --defaultAdd --filename admin-group.ldif
   ```

3. Create another LDIF file to add the access control instruction (ACI) to the root suffix, or base DN to provide full access to the Directory Server to the new administrator. Save the file as `admin-aci.ldif`.

   ```
   dn: dc=example,dc=com
   changetype: modify
   add: aci
   aci: (target="ldap:///dc=example,dc=com")
     (targetattr != "aci")
     (version 3.0; acl "allow all Admin group";
       allow(all) groupdn = "ldap:///cn=Dir
    Admins,ou=Groups,dc=example,dc=com";)
   ```

4. Then, add the ACI using the `ldapmodify` tool:

   ```
   $ bin/ldapmodify --filename admin-aci.ldif
   ```

5. Verify the additions using `ldapsearch`. The first command searches for the entry that contains `cn=Dir Admins` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including access control instructions, associated with the entry.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(cn=Dir Admins)"

   $ bin/ldapsearch --baseDN dc=example,dc=com --searchScope base \
     "(objectclass=*)" "+"
   ```

6. Add specific privileges to each Admin account using an LDIF file, saved as `admin-priv.ldif`. In this example, add the `password-reset` privilege to the `user.0` admin account from the command line. Add the privilege using the `ldapmodify` tool. Repeat the process for the other administrators configured in the Admin group.

   ```
   dn: uid=user.0,ou=People,dc=example,dc=com
   changetype: modify
   add: ds-privilege-name
   ds-privilege-name: password-reset

   $ bin/ldapmodify --filename admin-priv.ldif
   ```

   ```
   Processing MODIFY request for uid=user.0,dc=example,dc=com
   MODIFY operation successful for DN uid=user.0,dc=example,dc=com
   ```

7. Assign a password policy for the Admin account using an LDIF file, saved as `admin-pwd-policy.ldif`. For example, create an "Admin Password Policy", then add the password policy to the account. Apply the password policy to the account using the `ldapmodify` tool.

   ```
   dn: uid=user.0,dc=example,dc=com
   changetype: modify
   add: ds-pwp-password-policy-dn
   ds-pwp-password-policy-dn: cn=Admin Password Policy,cn=Password
    Policies,cn=config

   $ bin/ldapmodify --filename admin-pwd-policy.ldif
   ```

Configuring a Global Administrator

A global administrator is created when replication is enabled, and is responsible for managing configuration server groups. A configuration server group is an administration domain that allows you to synchronize configuration changes to one or all of the servers in the group. For example, you can set up a group when configuring a replication topology, where configuration changes to one server can be applied to all of the servers at a time.

Global Administrator(s) are stored in the topology registry. These entries are always mirrored between servers in a topology. Global Administrators can be assigned privileges like other admin users but are typically used to manage the data under `cn=topology,cn=config` and `cn=config`. You can create new global administrators and remove existing global administrators using the `dsconfig` tool. The global administrator entries are located in the `cn=Topology Admin User, cn=topology,cn=config` branch.

To Create a Global Administrator

1. Use `dsconfig` to create a new global administrator.

```
$ bin/dsconfig create-topology-admin-user \
  --user-name admin2 \
  --set alternate-bind-dn:cn=admin2 \
  --set password:rootPassword
```

2. To verify the creation of the new administrator, use the `list-topology-admin-users` subcommand with `dsconfig`.

```
$ bin/dsconfig list-topology-admin-users
Topology Admin User : Type
____________________:_______
admin               : generic
admin2              : generic
```

To Remove a Global Administrator

1. Use `dsconfig` to delete an existing global administrator.

```
$ bin/dsconfig delete-topology-admin-user --user-name admin2
```

2. To verify the deletion of the global administrator, use the list-topology-admin-users option with `dsconfig`.

```
$ bin/dsconfig list-topology-admin-users
Topology Admin User : Type
____________________:______
admin               : generic
```

Configuring Server Groups

The PingDirectory Server provides a mechanism for setting up administrative domains that synchronize configuration changes among servers in a server group. After you have set up a server group, you can make an update on one server using `dsconfig`, then you can apply the change to the other servers in the group using the --applyChangeTo server-group option of the `dsconfig` non-interactive command. If you want to apply the change to one server in the group, use the --applyChangeTo single-server option. When using `dsconfig` in interactive command-line mode, you will be asked if you want to apply the change to a single server or to all servers in the server group.

About the Server Group Example

You can create an administrative server group using the `dsconfig` tool. The general process is to create a group, add servers to the group, and then set a global configuration property to use the server group. If you are configuring a replication topology, then you must configure the replicas to be in a server group as outlined in Replication Configuration.

The following example procedure adds three Directory Server instances into the server group labelled "group-one".

To Create a Server Group

1. Create a group called “group-one” using `dsconfig`.

```
$ bin/dsconfig create-server-group --group-name group-one
```

2. Add any directory server to the server group. If you have set up replication between a set of servers, these server entries will have already been created by the `dsreplication` enable command.

```
$ bin/dsconfig set-server-group-prop --group-name group-one --set
 member:server1

$ bin/dsconfig set-server-group-prop --group-name group-one --set
 member:server2

$ bin/dsconfig set-server-group-prop --group-name group-one --set
 member:server3
```

3. Set a global configuration property for each of the servers that should share changes in this group.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:group-one
```

4. Test the server group. In this example, enable the log publisher for each directory server in the group, server-group, by using the --applyChangeTo server-group option.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true \
  --applyChangeTo server-group
```

5. View the property on the first directory server instance.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : true
```

6. Repeat step 5 on the second and third directory server instance.
7. Test the server group by disabling the log publisher on the first directory server instance by using the --applyChangeTo single-server.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:disabled \
  --applyChangeTo single-server
```

8. View the property on the first directory server instance. The first directory server instance should be disabled.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : false
```

9. View the property on the second directory server instance. Repeat this step on the third directory server instance to verify that the property is still enabled on that server.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : true
```

Configuring Client Connection Policies

Client connection policies help distinguish what portions of the DIT the client can access. They also enforce restrictions on what clients can do in the server. A client connection policy specifies criteria for membership based on information about the client connection, including client address, protocol, communication security, and authentication state and identity. The client connection policy, however, does not control membership based on the type of request being made.

Every client connection is associated with exactly one client connection policy at any given time, which is assigned to the client when the connection is established. The choice of which client connection policy to use will be reevaluated when the client attempts a bind to change its authentication state or uses the StartTLS extended operation to convert an insecure connection to a secure one. Any changes you make to the client connection policy do not apply to existing connections. The changes only apply to new connections.

Client connections are always unauthenticated when they are first established. If you plan to configure a policy based on authentication, you must define at least one client connection policy with criteria that match unauthenticated connections.

Once a client has been assigned to a policy, you can determine what operations they can perform. For example, your policy might allow only SASL bind operations. Client connection policies are also associated with one or more subtree views, which determine the portions of the DIT a particular client can access. For example, you might configure a policy that prevents users connecting over the extranet from accessing configuration information. The client connection policy is evaluated in addition to access control, so even a root user connecting over the extranet would not have access to the configuration information.

Understanding the Client Connection Policy

Client connection policies are based on two things:

- **Connection criteria**. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used in other instances when the server needs to perform matching based on connection-level properties, such as filtered logging. A single connection can match multiple connection criteria definitions.
- **Evaluation order index**. If multiple client connection policies are defined in the server, then each of them must have a unique value for the **evaluation-order-index** property. The client connection policies are evaluated in order of ascending evaluation order index. If a client connection does not match the criteria for any defined client connection policy, then that connection will be terminated.

If the connection policy matches a connection, then the connection is assigned to that policy and no further evaluation occurs. If, after evaluating all of the defined client connection policies, no match is found, the connection is terminated.

When a Client Connection Policy is Assigned

A client connection policy can be associated with a client connection at the following times:

- When the connection is initially established. This association occurs exactly once for each client connection.
- After completing processing for a StartTLS operation. This association occurs at most once for a client connection, because StartTLS cannot be used more than once on a particular connection. You also may not stop using StartTLS while keeping the connection active.

- After completing processing for a bind operation. This association occurs zero or more times for a client connection, because the bind request can be processed many times on a given connection.

StartTLS and bind requests will be subject to whatever constraints are defined for the client connection policy that is associated with the client connection at the time that the request is received. Once they have completed, then subsequent operations will be subject to the constraints of the new client connection policy assigned to that client connection. This policy may or may not be the same client connection policy that was associated with the connection before the operation was processed. That is, any policy changes do not apply to existing connections and will be applicable when the client reconnects.

All other types of operations will be subject to whatever constraints are defined for the client connection policy used by the client connection at the time that the request is received. The client connection policy assigned to a connection never changes as a result of processing any operation other than a bind or StartTLS. So, the server will not re-evaluate the client connection policy for the connection in the course of processing an operation. For example, the client connection policy will never be re-evaluated for a search operation.

Restricting the Type of Search Filter Used by Clients

You can restrict the types of search filters that a given client may be allowed to use to prevent the use of potentially expensive filters, like range or substring searches. You can use the `allowed-filter-type` property to provide a list of filter types that may be included in the search requests from clients associated with the client connection policy. This setting will only be used if search is included in the set of allowed operation types. This restriction will only be applied to searches with a scope other than `baseObject`, such as searches with a scope of `singleLevel`, `wholeSubtree`, or `subordinateSubtree`.

The `minimum-substring-length` property can be used to specify the minimum number of non-wildcard characters in a substring filter. Any attempt to use a substring search with an element containing fewer than this number of bytes will be rejected. For example, the server can be configured to reject filters like `"(cn=a*)"` and `"(cn=ab*)"`, but to allow `"(cn=abcde*)"`. This property setting will only be used if search is included in the set of allowed operation types and at least one of `sub-initial`, `sub-any`, or `sub-final` is included in the set of allowed filter types.

There are two primary benefits to enforcing a minimum substring length:

- Allowing very short substrings can require the server to perform more expensive processing. The search requires a lot more server effort to assemble a candidate entry list for short substrings because the server has to examine a lot more index keys.
- Allowing very short substrings makes it easier for a client to put together a series of requests to retrieve all the data from the server (a process known as "trawling"). If a malicious user wants to obtain all the data from the server, then it is easier to issue 26 requests like `"(cn=a*)"`, `"(cn=b*)"`, `"(cn=c*)"`, ..., `"(cn=z*)"` than if the user is required to do something like `"(cn=aaaaa*)"`, `"(cn=aaaab*)"`, `"(cn=aaaac*)"`, ..., `"(cn=zzzzz*)"`.

Setting Resource Limits

Client connection policies can specify resource limits, helping to ensure that no single client monopolizes server resources. You can limit the total number of connections to a server from a particular client or from clients that match specified criteria. You can also limit the duration of the connection.

A client connection policy may only be used to enforce additional restrictions on a client connection. You can never use it to grant a client capabilities that it would not otherwise have.

Any change to any of these new configuration properties will only impact client connections that are assigned to the client connection policy after the change is made. Any connection associated with the client connection policy before the configuration change was made will continue to be subject to the configuration that was in place at the time it was associated with that policy.

Table 6: Resource Limiting Properties

Property	Description
maximum-concurrent-connections	Specifies the maximum number of client connections that can be associated with that client connection policy at any given time. The default value of zero indicates that no limit will be enforced. If the server already has the maximum number of connections associated with a client connection policy, then any attempt to associate another connection with that policy (e.g., newly-established connections or an existing connection that has done something to change its client connection policy, such as perform a bind or StartTLS operation) will cause that connection to be terminated.
terminate-connection	Specifies that any client connection for which the client connection policy is selected (whether it is a new connection or an existing connection that is assigned to the client connection policy after performing a bind or StartTLS operation) will be immediately terminated. This property can be used to define criteria for connections that you do not want to be allowed to communicate with the Directory Server.
maximum-connection-duration	Specifies the maximum length of time that a connection associated with the client connection policy can remain established to the Directory Server, regardless of the amount of activity on that connection. A value of "0 seconds" (default) indicates that no limit will be enforced. If a connection associated with the client connection policy has been established for longer than this time, then it will be terminated.
maximum-idle-connection-duration	Specifies the maximum length of time that a connection associated with the client connection policy can remain established with the Directory Server without any requests in progress. A value of "0 seconds" (default) indicates that no additional limit will be enforced on top of whatever idle time limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective idle time limit for any client connection will be the smaller of the `maximum-idle-connection-duration` from the client connection policy and the idle time limit that would otherwise be in effect for that client. This property can be used to apply a further restriction on top of any value that may be enforced by the `idle-time-limit` global configuration property (which defines a default idle time limit for client connections) or the `ds-rlim-idle-time-limit` operational attribute (which may be included in a user entry to override the default idle time limit for that user).
maximum-operation-count-per-connection	Specifies the maximum number of operations that a client associated with the client connection policy will be allowed to request. A value of zero (default) indicates that no limit will be enforced. If a client attempts to request more than this number of operations on the same connection, then that connection will be terminated.
maximum-concurrent-operations-per-connection	Specifies the maximum number of operations that may be active at any time from the same client. This limit is only applicable to clients that use asynchronous operations with multiple outstanding requests at any given time. A value of zero (default) indicates that no limit will be enforced.

Property	Description
	If a client already has the maximum number of outstanding requests in progress and issues a new request, then that request will be delayed and/or rejected based on the value of the `maximum-concurrent-operation-wait-time-before-rejecting` property.
maximum-concurrent-operation-wait-time-before-rejecting	Specifies the maximum length of time that a client connection should allow an outstanding operation to complete if the maximum number of concurrent operations for a connection are already in progress when a new request is received on that connection. A value of “0 seconds” (default) indicates that any new requests received while the maximum number of outstanding requests are already in progress for that connection will be immediately rejected. If an outstanding operation completes before this time expires, then the server may be allowed to process that operation. If the time expires, the new request will be rejected.
maximum-ldap-join-size-limit	Specifies the maximum number of entries that can be directly joined with any individual search result entry. A value of zero indicates that no LDAP join size limit is enforced. The limit can be overridden on a per-user basis using the `ds-rlim-ldap-join-size-limit` operational attribute. The LDAP join size limit is also restricted by the search operation size limit. If a search result entry is joined with more entries than allowed, the join result control will have a "size limit exceeded" (integer value 4) result code.
allowed-request-control	Specifies the OIDs of the request controls that clients associated with the client connection policy will be allowed to use. If any allowed-request-control OIDs are specified, then any request which includes a control not in that set will be rejected. If no `allowed-request-control` values are specified (default), then any control whose OID is not included in the set of denied-request-control values will be allowed.
denied-request-control	Specifies the OIDs of the request controls that clients associated with the client connection policy will not be allowed to use. If there are any `denied-request-control` values, then any request containing a control whose OID is included in that set will be rejected. If there are no `denied-request-control` values (default), then any request control will be allowed if the `allowed-request-control` property is also empty, or only those controls whose OIDs are included in the set of `allowed-request-control` values will be allowed if at least one `allowed-request-control` value is provided.
allowed-filter-type	Specifies the types of components which may be used in filters included in search operations with a non-base scope that are requested by clients associated with the client connection policy. Any non-base scoped search request whose filter contains a component not included in this set will be rejected. The set of possible filter types include: and or not equality sub-initial sub-any

Property	Description
	sub-final greater-or-equal less-or-equal approximate-match extensible-match By default, all filter types will be allowed. Also note that no restriction will be placed on the types of filters which may be used in searches with a base scope.
allow-unindexed-searches	Specifies whether clients associated with the client connection policy will be allowed to request searches which cannot be efficiently processed using the configured set of indexes. Note that clients will still be required to have the `unindexed-search` privilege, so this option will not grant the ability to perform unindexed searches to clients that would not have otherwise had that ability, but it may be used to prevent clients associated with the client connection policy from requesting unindexed searches when they might have otherwise been allowed to do so. By default, this has a value of "true", indicating that any client associated with the client connection policy that has the `unindexed-search` privilege will be allowed to request unindexed searches.
minimum-substring-length	Specifies the minimum number of bytes, which may be present in any sub-Initial, subAny, or subFinal element of a substring search filter component in a search with a non-baseObject scope. A value of one (which is the default) indicates that no limit will be enforced. This property may be used to prevent clients from issuing overly-vague substring searches that may require the Installing the Directory Server to examine too many entries over the course of processing the request.
maximum-search-size-limit	Specifies the maximum number of entries that may be returned from any single search operation requested by a client associated with this client connection policy. Note that this property only specifies a maximum limit and will never increase any limit that may already be in effect for the client via the `size-limit` global configuration property or the `ds-rlim-size-limit` operational attribute. A value of zero (default) indicates that no additional limit will be enforced on top of whatever size limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective maximum size limit for any search operation requested by the client will be the smaller of the size limit from that search request, the `maximum-search-size-limit` from the client connection policy, and the size limit that would otherwise be in effect for that client.

Defining the Operation Rate

You can configure the maximum operation rate for individual client connections as well as collectively for all connections associated with a client connection policy. If the operation rate limit is exceeded, the Directory Server may either reject the operation or terminate the connection. You can define multiple rate limit values, making it possible to fine tune limits for both a long term average operation rate and short term operation bursts. For example, you can define a limit of one thousand operations per second and one million operations per day, which works out to an average of less than twelve operations per second, but with bursts of up to one thousand operations per second.

Rate limit strings should be specified as a maximum count followed by a slash and a duration. The count portion must contain an integer, and may be followed by a multiplier of k (to indicate that the integer should be interpreted as thousands), m (to indicate that the integer should be interpreted as millions), or g (to indicate that the integer should be interpreted as billions). The duration portion must contain a time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), and may be preceded by an integer to specify a quantity for that unit.

For example, the following are valid rate limit strings:

1/s (no more than one operation over a one-second interval)
10K/5h (no more than ten thousand operations over a five-hour interval)
5m/2d (no more than five million operations over a two-day interval)

You can provide time units in many different formats. For example, a unit of seconds can be signified using s, sec, sect, second, and seconds.

Client Connection Policy Deployment Example

In this example scenario, we assume the following:

Two external LDAP clients are allowed to bind to the Directory Server.
Client 1 should be allowed to open only 1 connection to the server.
Client 2 should be allowed to open up to 5 connections to the server.

Defining the Connection Policies

We need to set a per-client connection policy limit on the number of connections that may be associated with a particular client connection policy. We have to define at least two client connection policies, one for each of the two clients. Each policy must have different connection criteria for selecting the policy with which a given client connection should be associated.

Because the criteria is based on authentication, we must create a third client connection policy that applies to unauthenticated clients, because client connections are always unauthenticated as soon as they are established and before they have sent a bind request. Plus, clients are not required to send a bind request as their first operation.

Therefore, we define the following three client connection policies:

Client 1 Connection Policy, which only allows client 1, with an evaluation order index of 1.
Client 2 Connection Policy, which only allows client 2, with an evaluation order index of 2.
Unauthenticated Connection Policy, which allows unauthenticated clients, with an evaluation order index of 3.

We define simple connection criteria for the Client 1 Connection Policy and the Client 2 Connection Policy with the following properties:

The `user-auth-type` must not include none, so that it will only apply to authenticated client connections.

The `included-user-base-dn` should match the bind DN for the target user. This DN may be full DN for the target user, or it may be the base DN for a branch that contains a number of users that you want treated in the same way.

To create more generic criteria that match more than one user, you could list the DNs of each of the users explicitly in the `included-user-base-dn` property. If there is a group that contains all of the pertinent users, then you could instead use the `[all|any|not-all|not-any]-included-user-group-dn` property to apply to all members of that group. If the entries for all of the users match a particular filter, then you could use the `[all|any|not-all|not-any]-included-user-filter` property to match them.

How the Policy is Evaluated

Whenever a connection is established, the server associates the connection with exactly one client connection policy. The server does this by iterating over all of the defined client connection policies in ascending order of the evaluation order index. Policies with a lower evaluation order index value will be examined before those with a higher evaluation order index value. The first policy that the server finds whose criteria match the client connection will be associated with that connection. If no client connection policy is found with criteria matching the connection, then the connection will be terminated.

So, in our example, when a new connection is established, the server first checks the connection criteria associated with the Client 1 Connection Policy because it has the lowest evaluation order index value. If it finds that the criteria do not match the new connection, the server then checks the connection criteria associated with the Client 2 Connection Policy because it has the second lowest evaluation order index. If these criteria do not match, the server finally checks the connection criteria associated with the Unauthenticated Connection Policy, because it has the third lowest evaluation order index. It finds a match, so the client connection is associated with the Unauthenticated Connection Policy.

After the client performs a bind operation to authenticate to the server, then the client connection policies will be re-evaluated. If client 2 performs the bind, then the Client 1 Connection Policy will not match but the Client 2 Connection Policy will, so the connection will be re-associated with that client connection policy. Whenever a connection is associated with a client connection policy, the server will check to see if the maximum number of client connections have already been associated with that policy. If so, then the newly-associated connection will be terminated.

For example, Client 1 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. Client 1 then sends a bind request. The determination of whether the bind operation is allowed is made based on the constraints defined in the Unauthenticated Connection Policy, because it is the client connection policy already assigned to the client connection. Once the bind has completed, then the server will reevaluate the client connection policy against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to the Client 1 Connection Policy.

Next, Client 2 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. When Client 2 sends a bind request, the operation is allowed based on the constraints defined in the Unauthenticated Connection Policy. Once the bind is complete, the client connection is evaluated against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria do not match, so the client 2 connection is evaluated against the connection criteria associated with Client 2 Connection Policy, because it has the next lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to Client 2 Connection Policy.

Client 1 sends a search request. The Client 1 Connection Policy is used to determine whether the search operation should be allowed, because this is the client connection policy assigned to the client connection for client 1. The connection is not reevaluated, before or after processing the search operation.

To Configure a Client Connection Policy Using the Console

1. Open the Administrative Console. Provide a username and password, and then click **Login**.
2. In the **Core Server** section, click **Client Connection Policies**. If you do not see **Client Connection Policies** on the menu, change the Object Types filter to **Standard**.
3. Click **Add New** to add a new policy.
4. Enter a Policy ID. If you want to base your new client connection policy on an existing policy, select it from the **Template** menu.
5. Configure the properties of the client connection policy. To enable the policy, select **Enabled**.
6. Enter the order in which you want the new policy to be evaluated in the **Evaluation Order Index** box, and then click **Continue**. A policy with a lower index is evaluated before a policy with a higher index. The Directory Server uses the first evaluated policy that applies to a client connection.
7. Select the connection criteria that match the client connection for this policy. Click **View and edit** to change the criteria. Click **Select New** to add new criteria. Select the operations allowed for clients that are members of this connection group. Use the **Add and Remove** buttons to make operations available to clients. Specify the extended operations that clients are allowed and denied to use.
8. Enter the type of authorization allowed and the SASL mechanisms that are allowed and denied in response to client requests.
9. Check the **Include Backend Subtree Views** check box if you want to automatically include the subtree views of backends configured in the Directory Server. You can also choose to include and exclude specific base DNs using the appropriate fields.

10. Once you have finished configuring the properties of your client connection policy, click **Confirm then Save** to review the `dsconfig` command equivalent and save your changes. Click **Save Now** to save your changes without first reviewing the `dsconfig` output.

To Configure a Client Connection Policy Using dsconfig

You can configure a client connection policy using the `dsconfig` tool in interactive mode from the command line. You can access the **Client Connection Policy** menu on the **Standard objects** menu.

1. Use the `dsconfig` tool to create and configure a client connection policy. Specify the host name, connection method, port number, and bind DN as described in previous procedures.
2. On the Directory Server main menu, change to the **Standard objects** menu by entering `o` and then entering the number for the Standard menu.
3. On the Directory Server main menu, enter the number associated with **Client Connection Policy**.
4. On the **Client Connection Policy management** menu, type the number corresponding to **Create a new connection policy**.
5. Enter `n` to create a new client connection policy from scratch.
6. Next, enter a name for the new client connection policy.
7. On the **Enabled Property** menu, select true to enable the connection policy.
8. On the **Evaluation-Order Property** menu, type a value between 0 and 2147483647 to set the evaluation order for the policy. A client connection policy with a lower evaluation-order will be evaluated before one with a higher number. For this example, type `9999`.
9. On the **Client Connection Policy** menu, review the configuration. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the client connection policy.

   Any changes that you make to the client connection policy do not apply to existing connections. They will only apply to new connections.

Restricting Server Access Based on Client IP Address

Another common use case is to limit client access to the Directory Server. Two methods are available:

- **Connection Handlers**. You can limit the IP addresses using the LDAP or LDAPS connection handlers. The connection handlers provide an `allowed-client` property and a `denied-client` property. The `allowed-client` property specifies the set of allowable address masks that can establish connections to the handler. The `denied-client` property specifies the set of address masks that are not allowed to establish connections to the handler.
- **Client Connection Policies**. You can take a more fine-grained approach by restricting access by configuring a new Client Connection Policy, then create a new connection criteria and associate it with the connection policy. Connection criteria define sets of criteria for grouping and describing client connections based on a number of properties, including the protocol, client address, connection security, and authentication state for the connection. Each client connection policy may be associated with zero or more Connection Criteria, and server components may use Connection Criteria to indicate which connections should be processed and what kind of processing should be performed (e.g., to select connections and/or operations for filtered logging, or to classify connections for network groups).

To Restrict Server Access Using the Connection Handlers

- Use `dsconfig` to set the `allowed-client` property for the LDAP connection handler. You should specify the address mask for the range of allowable IP addresses that can establish connections to the Directory Server. You should also specify the loopback address, 127.0.0.1, so that you will still be able to configure the server using the `dsconfig` tool on the local host.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --set "allowed-client:10.6.1.*" \
  --set allowed-client:127.0.0.1
```

To Restrict Server Access Using Client Connection Policies

1. Create a simple connection criteria. The following example uses the `dsconfig` tool in non-interactive mode. It allows only the Directory Server's IP address and loopback to have access.

   ```
   $ bin/dsconfig set-connection-criteria-prop \
     --criteria-name allowed-ip-addrs \
     --add included-client-address:10.6.1.80 \
     --add included-client-address:127.0.0.1
   ```

2. Assign the criteria to the client connection policy. After you have run the following command, access is denied to remote IP addresses. The Directory Server does not require a restart.

   ```
   $ bin/dsconfig set-client-connection-policy-prop \
     --policy-name new-policy \
     --set connection-criteria:allowed-ip-addrs
   ```

3. Add a remote IP range to the criteria. For this example, add 10.6.1.*. Access from any remote servers is allowed. The Directory Server does not require a restart.

   ```
   $ bin/dsconfig set-connection-criteria-prop \
     --criteria-name allowed-ip-addrs \
     --add "included-client-address:10.6.1.*"
   ```

4. To restore default behavior, you can remove the criteria from the connection policy. The Directory Server does not require a restart. Remember to include the LDAP or LDAPS connection parameters (e.g., hostname, port, bindDN, bindPassword) with the `dsconfig` command.

   ```
   $ bin/dsconfig set-client-connection-policy-prop \
     --policy-name new-policy --remove connection-criteria:allowed-ip-addrs
   ```

To Automatically Authenticate Clients that Have a Secure Communication Channel

The Directory Server provides an option to automatically authenticate clients that have a secure communication channel (either SSL or StartTLS) and presented their own certificate. This option is disabled by default, but when enabled, the net effect will be as if the client issued a SASL EXTERNAL bind request on that connection.

This option will be ignored if the client connection is already authenticated (e.g., because it is using StartTLS but the client had already performed a bind before the StartTLS request). If the bind attempt fails, then the connection will remain unauthenticated but usable. If the client subsequently sends a bind request on the connection, then it will be processed as normal and any automatic authentication will be destroyed.

- Run the following `dsconfig` command.

  ```
  $ bin/dsconfig set-connection-handler-prop \
    --handler-name "LDAPS Connection Handler" \
    --set "auto-authenticate-using-client-certificate:true"
  ```

Securing the Server with Lockdown Mode

The Directory Server provides tools to enter and leave lockdown mode if the server requires a security lockdown. In lockdown mode, only users with the `lockdown-mode` privilege can perform operations, while those without the privilege are rejected. Root users have this privilege by default; other administrators can be given this privilege.

The Directory Server can be manually placed into lockdown mode to perform some administrative operation while ensuring that other client requests are not allowed to access any data in the server. In addition, some configuration problems (particularly problems that could lead to inadvertent exposure of sensitive information, like an access control rule that cannot be properly parsed) cause the server to place itself in lockdown mode, so that an administrator can manually correct the problem. Lockdown mode does not persist across restarts. The directory server can be taken out of lockdown mode by using either the `leave-lockdown-mode` tool or by restarting the server. If administrators want to start a server in lockdown mode, they can use the `start-server` --lockdownMode option.

Any client request to the Directory Server in lockdown mode receives an "Unavailable" response.

To Manually Enter Lockdown Mode

- Use `enter-lockdown-mode` to begin a lockdown mode.

```
$ bin/enter-lockdown-mode
```

To Leave Lockdown Mode

- Use `leave-lockdown-mode` to end lockdown mode.

```
$ bin/leave-lockdown-mode
```

To Start a Server in Lockdown Mode

- Use the --lockdownMode option with the `start-server` tool to start a server in lockdown mode.

```
$ bin/start-server --lockdownMode
```

Configuring Maximum Shutdown Time

During shutdown, some database checkpointing and cleaning threads may remain active even after the default time period on systems with very large or very busy database backends. If checkpointing or cleaning is aborted prematurely, it could possible lead to significantly longer startup times for the Directory Server. The Directory Server provides an option for administrators to set the maximum time a shutdown process should take. When a shutdown process is initiated, the server begins stopping all of its internal components and waits up to 5 minutes for all threads to complete before exiting.

Administrators can use the `dsconfig` tool to increase the maximum shutdown time to allow database operations to complete.

To Configure the Maximum Shutdown Time

- Use the `dsconfig` tool to increase the maximum shutdown time for your system. The following command increases the maximum shutdown time from 5 minutes to 6 minutes. The command allows time values of w (weeks), d (days), h (hours), m (minutes), s (seconds), ms (milliseconds).

  ```
  $ bin/dsconfig set-global-configuration-prop --set "maximum-shutdown-time:6
   m"
  ```

  Remember to include the LDAP or LDAPS connection parameters (e.g., host name, port, bindDN and bindDN password) with the `dsconfig` command.

  The `maximum-shutdown-time` property can also be changed using the `dsconfig` tool in interactive mode. From the main menu, select **Global Configuration**, and then select the option to display advanced properties.

Working with Referrals

A referral is a redirection mechanism that tells client applications that a requested entry or set of entries is not present on the Directory Server but can be accessed on another server. Referrals can be used when entries are located on another server. The Directory Server implements two types of referrals depending on the requirement.

- **Referral on Update Plug-in**. The Directory Server provides a Referral on Update Plug-in to create any referrals for update requests (add, delete, modify, or modDN operations) on read-only servers. For example, given two replicated directory servers where one server is a master (read-write) and the other, a read-only server, you can configure a referral for any client update requests on the second directory server to point to the master server. If a client application sends an add request, for example, on the second directory server, the directory server responds with a referral that indicates any updates should be made on the master server. All read requests on the read-only server will be processed as normal.

- **Smart Referrals**. The Directory Server supports smart referrals that map an entry or a specific branch of a DIT to an LDAP URL. Any client requests (reads or writes) targeting at or below the branch of the DIT will send a referral to the server designated in the LDAP URL.

Specifying LDAP URLs

Referrals use LDAP URLs to redirect a client application's request to another server. LDAP URLs have a specific format, described in RFC 4516 and require that all special characters be properly escaped and any spaces indicated as "%20". LDAP URLs have the following syntax:

```
ldap[s]://hostname:port/base-dn?attributes?scope?filter
```

where

- *ldap[s]* indicates the type of LDAP connection to the Directory Server. If the Directory Server connects over a standard, non-encrypted connection, then ldap is used; if it connects over SSL, then ldaps is used. Note that any search request initiated by means of an LDAP URL is anonymous by default, unless an LDAP client provides authentication.
- *hostname* specifies the host name or IP address of the Directory Server.
- *port* specifies the port number of the Directory Server. If no port number is provided, the default LDAP port (389) or LDAPS port (636) is used.
- *base-dn* specifies the distinguished name (DN) of an entry in the DIT. The Directory Server uses the base DN as the starting point entry for its searches. If no base DN is provided, the search begins at the root of the DIT.
- *attributes* specifies those attributes for which the Directory Server should search and return. You can indicate more than one attribute by providing a comma-separated list of attributes. If no attributes are provided, the search returns all attributes.
- *scope* specifies the scope of the search, which could be one of the following: `base` (only search the specified base DN entry), `one` (only search one level below the specified base DN), `sub` (search the base entry and all entries below the specified base DN). If no scope is provided, the server performs a base search.
- *filter* specifies the search filter to apply to entries within the scope of the search. If no filter is provided, the server uses +.

Creating Referrals

You can create a smart referral by adding an entry with the referral and `extensibleObject` object classes or adding the object classes to a specific entry. The referral object class designates the entry as a referral object. The `extensibleObject` object class allows you to match the target entry by matching any schema attribute. The following example shows how to set up a smart referral if a portion of a DIT is located on another server.

To Create a Referral

1. Create an LDIF file with an entry that contains the `referral` and `extensibleObject` object classes.

```
dn: ou=EngineeringTeam1,ou=People,dc=example,dc=com
objectClass: top
objectClass: referral
objectClass: extensibleObject
ou: Engineering Team1
ref: ldap://server2.example.com:6389/
ou=EngineeringTeam1,ou=People,dc=example,dc=com
```

2. On the first server, add the referral entry using the `ldapmodify` command.

```
$ bin/ldapmodify --defaultAdd --fileName referral-entry.ldif
```

3. Verify the addition by searching for a user.

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com "(uid=user.4)"
```

```
SearchReference(referralURLs={ldap://server2.example.com:6389/
 ou=EngineeringTeam1,ou=People,dc=example,dc=com??sub?})
```

To Modify a Referral

- Use `ldapmodify` with the `manageDSAIT` control to modify the `ref` attribute on the referral entry.

  ```
  $ bin/ldapmodify --control manageDSAIT
  dn: ou=EngineeringTeam1,ou=People,dc=example,dc=com
  changetype: modify
  replace: ref
  ref: ldap://server3.example.com/
  ou=EngineeringTeam1,ou=People,dc=example,dc=com
  ```

To Delete a Referral

- Use `ldapdelete` with the `manageDSAIT` control to delete the referral entry.

  ```
  $ bin/ldapdelete \
    --control manageDSAIT "ou=EngineeringTeam1,ou=People,dc=example,dc=com"
  ```

Configuring a Read-Only Server

The PingDirectory Server provides a means to configure a hub-like, read-only directory server for legacy systems that require it. The read-only directory server participates in replication but cannot respond to any update requests from an external client. You can configure the Directory Server by setting the writability mode to internal-only, which makes the server operate in read-only mode. Read-only mode directory servers can process update operations from internal operations but reject any write requests from external clients. Because the Directory Server cannot accept write requests, you can configure the server to send a referral, which redirects a client's request to a master server. The client must perform the operation again on the server named in the referral.

Note: For Implementers of Third Party Extensions. Many Server SDK extensions use the `InternalConnection` interface to process operations in the server, rather than issuing LDAP requests over the network. If an extension does so in response to an external update request, then any Directory Server using that extension will effectively respond to external update requests, even though the Directory Server is configured to operate in read-only mode, as described above. One possible workaround is to split the extension into two extensions, one for reads and one for writes, then disabling (or not deploying) the write-only extension when configuring a Directory Server in read-only mode.

To Configure a Read-Only Server

1. Install two replicating directory servers. See Replication Configuration for various ways to set up your servers.
2. On the second server, use the `dsconfig` command to set the writability mode of the server to internal-only.

   ```
   $ bin/dsconfig set-global-configuration-prop \
     --set writability-mode:internal-only
   ```
3. On the second server, use the `dsconfig` tool to create a referral that instructs the server to redirect client write requests under `dc=example,dc=com` to server1.example.com:1389. The referral itself is defined as a plugin of type `Referral on Update`. This command sets up the server to process read operations but redirects all write operations under `dc=example,dc=com` to another server.

   ```
   $ bin/dsconfig create-plugin --plugin-name "Refer Updates" \
     --type referral-on-update \
     --set enabled:true \
     --set referral-base-url:ldap://server1.example.com:1389/ \
     --set "base-dn:dc=example,dc=com"
   ```

4. To test the referral, attempt to modify an entry and confirm that the server responds with the result code of 10. The resulting message is available in the server's access log.

```
$ bin/ldapmodify -p 2389 -D "cn=Directory Manager" -w password
dn: uid=user.12,ou=People,dc=example,dc=com
changetype:modify
replace:telephoneNumber
telephoneNumber: +1 408 555 1155
```

```
[06/Aug/2012:15:28:21.468 -0400] MODIFY
RESULT conn=86 op=1 msgID=1 requesterIP="127.0.0.1"
dn="uid=user.12,ou=People,dc=example,dc=com" resultCode=10
referralURLs="ldap://server1.example.com:1389/uid=user.12,
ou=People,dc=example,dc=com" etime=0.223
```

Configuring HTTP Access for the Directory Server

Although most clients communicate with the PingDirectory Server using LDAP, the server also provides support for an HTTP connection handler that uses Java servlets to serve content to clients over HTTP. Ping Identity offers an extension that uses this HTTP connection handler to add support for the System for Cross-domain Identity Management (SCIM) protocol. Third-party developers can also use the Server SDK to write extensions that leverage this HTTP support.

The following sections describe how to configure HTTP servlet extensions and how to configure an HTTP connection handler.

Configuring HTTP Servlet Extensions

To use the HTTP connection handler, you must first configure one or more servlet extensions. Servlet extensions are responsible for obtaining Java servlets (using the Java Servlet 3.0 specification as described in JSR 315) and registering them to be invoked using one or more context paths. If you plan to deploy the SCIM extension, then you should follow the instructions in Chapter 24, "*Managing the SCIM Servlet Extension*." For custom servlet extensions created using the Server SDK, the process varies based on whether you are using a Java-based or Groovy-scripted extension.

Web Application Servlet Extensions

A Web application may be deployed either as a WAR file that has been packaged according to the standard layout and containing a web.xml deployment descriptor, or from a directory containing the application's source components arranged in the standard layout.

When deploying a Web application from a directory, you may specify the location of the `web.xml` deployment descriptor if it is not in the standard location. You may also specify the directory used by the server for temporary files. At runtime the web application has access to the server classes.

Java-based Servlet Extensions

For Java-based extensions, first use the Server SDK to create and build the extension bundle as described in the Server SDK documentation. Then, install it using the `manage-extension` tool as follows:

```
$ bin/manage-extension --install/path/to/extension.zip
```

The Java-based extension may then be configured for use in the server using `dsconfig` or the Administrative Console. Create a new Third Party HTTP Servlet Extension, specifying the fully-qualified name for the `HTTPServletExtension` subclass in the `extension-class` property, and providing any appropriate arguments in the `extension-argument` property.

👉 **Note:** Web Application and Servlet extensions run in a shared embedded web application server environment. Incompatiblities or conflicts may arise from use of different versions of commonly used jars or including frameworks such as loggers, Spring components, JAX-RS implementations or other resources which may require a dedicated Java Virtual Machine (JVM) environment. After introducing a custom extension, check the server error log for an indication that the extension loaded successfully. The error log may also contain debug information if the extension failed to load with an initialization exception or did not complete initialization.

Groovy-Scripted Extensions

For Groovy-scripted extensions, place the necessary Groovy scripts in the appropriate directory (based on the package for those scripts) below the `lib/groovy-scripted-extensions` directory. Then, create a new Groovy Scripted HTTP Servlet Extension, specifying the fully-qualified Groovy class name for the `script-class` property, and providing any appropriate arguments in the `script-argument` property.

Configuring HTTP Operation Loggers

Servlet extensions may write error log messages in the same way as any other kind of server component, but interaction with HTTP clients will not be recorded in the server access log. However, if a servlet extension performs internal operations to interact with data held in the directory server, then those operations may be captured in the access log. To capture information about communication with HTTP clients, you must configure one or more HTTP operations loggers.

By default, the server comes with a single HTTP operation logger implementation, which uses the standard W3C common log format. It records messages in a format like the following:

```
127.0.0.1 - - [01/Jan/2012:00:00:00 -0600]"GET/hello HTTP/1.1" 200 113
```

The log message contains the following elements:

- The IP address of the client that issued the request.
- The RFC 1413 (ident) identity of the client. Because the ident protocol is not typically provided by HTTP clients, the HTTP connection handler never requests this information. This identity will always be represented as a dash to indicate that information is not available.
- The authenticated identity determined for the request by HTTP authentication, or a dash to indicate that the request was not authenticated.
- The time that the request was received.
- The request issued by the client, including the HTTP method, path and optional query string, and the HTTP protocol version used.
- The integer representation of the HTTP status code for the response to the client.
- The number of bytes included in the body of the response to the client.

To configure an HTTP operation logger to use this common log format, create a new instance of a Common Log File HTTP Operation Log Publisher object, specifying the path and name for the active log file to be written and the rotation and retention policies that should be used to manage the log files. In general, properties for Common Log File HTTP Operation Log Publisher objects have the same meaning and use as they do for other kinds of loggers.

You can use the Server SDK to create custom Java-based or Groovy-scripted HTTP operation loggers using the Third Party HTTP Operation Log Publisher and Groovy Scripted HTTP Operation Log Publisher object types.

Example HTTP Log Publishers

When troubleshooting HTTP Connection Handler issues, administrators should first look at the logs to determine any potential problems. The following section shows some `dsconfig` commands and their corresponding log files.

Default Configuration Example. You can configure a default detailed HTTP Log Publisher with default log rotation and retention policies as follows:

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Detailed Access Logger" \
  --type detailed-http-operation \
  --set enabled:true \
  --set log-file:logs/http-detailed-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

The corresponding log file provides access information below. The log message contains the following elements:

The time that the request was received.
The request ID issued by the client, including the IP address, port, HTTP method, and URL.
The authorization type, request content type, and status code.
The response content length.
The redirect URI.
The response content type.

The HTTP log file is shown as follows:

```
[23/Feb/2012:01:19:45 -0600] RESULT requestID=4300604 from="10.5.1.10:53269"
method="GET" url="https://10.5.1.129:443/Gimel/Users/
uid=user.402914,ou=People,
dc=gimel" authorizationType="Basic" requestContentType="application/json"
statusCode=200 etime=4.145 responseContentLength=1530 redirectURI="https://
x2270-11.example.lab:443/Gimel/Users/uid=user.402914,ou=people,dc=gimel"
responseContentType="application/json"
[23/Feb/2012:01:19:45 -0600] RESULT requestID=4300605 from="10.5.1.10:53269"
method="PUT" url="https://10.5.1.129:443/Gimel/Users/
uid=user.207585,ou=people,
dc=gimel" authorizationType="Basic" requestContentType="application/json"
statusCode=200 etime=4.872 responseContentLength=1532 redirectURI="https://
x2270-11.example.lab:443/Gimel/Users/uid=user.207585,ou=people,dc=gimel"
responseContentType="application/json"
[23/Feb/2012:11:31:18 -0600] RESULT requestID=4309872 from="10.5.1.10:3
```

Configuration with Request/Response Header Names and Values. The following example adds request/response header names and values, including the "Content-Type" request header, which is normally suppressed by default.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "HTTP Detailed Access Logger" \
  --set log-request-headers:header-names-and-values \
  --remove suppressed-request-header-name:Content-Type \
  --set log-response-headers:header-names-and-values
```

The following is a log example of a query request by a SCIM Server using the property, `scim-query-rate`. The log message contains the basic log elements shown in the first example plus the following additional information:

The request header for the host, http method, content type, connection, user agent.
The response header for the access-control credentials.

```
[23/Oct/2012:11:39:41-0600] RESULT requestID=4665307 from="10.5.0.20:56044"
method="GET" url="https://10.5.1.129:443/Beth/Users?attributes=userName,title,
emails,urn:scim:schemas:extension:custom:1.0:descriptions,urn:scim:schemas:
extension:enterprise:1.0:manager,groups,urn:scim:schemas:extension:custom:1.0:
blob&filter=userName+eq+%22user.18935%22" requestHeader="Host:
 x2270-11.example.
lab:443" requestHeader="Accept: application/json" requestHeader="Content-Type:
application/json" requestHeader="Connection: keep-alive"
requestHeader="User-Agent: Wink Client v1.1.2" authorizationType="Basic"
requestContentType="application/json" statusCode=200 etime=140.384
responseContentLength=11778 responseHeader="Access-Control-Allow-Credentials:
true" responseContentType="application/json"
```

Another log example shows an example user creation event. The client is curl.

```
[23/Oct/2016:11:50:11-0600] RESULT requestID=4802791 from="10.8.1.229:52357"
method="POST" url="https://10.2.1.113:443/Aleph/Users/" requestHeader="Host:
 x2270-
11.example.lab" requestHeader="Expect: 100-continue" requestHeader="Accept:
 applica-
tion/xml" requestHeader="Content-Type: application/xml" requestHeader="User-
Agent:
curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4 OpenSSL/0.9.8r
 zlib/1.2.5"
authorizationType="Basic" requestContentType="application/xml" requestContent-
Length=1773 statusCode=201 etime=11.598 responseContentLength=1472 redirec-
tURI="https://x2270-11.example.lab:443/Aleph/Users/b2cef63c-5e46-11e1-974b-
60334b1a0d7a" responseContentType="application/xml"
```

The final example shows a user deletion request. The client is the Sync Server.

```
[23/Oct/2016:11:38:06-0600] RESULT requestID=4610261 from="10.5.1.114:34558"
method="DELETE" url="https://10.2.1.113:443/Aleph/Users/b8547525-24e0-41ae-
b66b-
0b441800de70" requestHeader="Host: x2270-11.example.lab:443"
 requestHeader="Accept:
application/json" requestHeader="Content-Type: application/json"
 requestHeader="Con-
nection: keep-alive" requestHeader="User-Agent: PingDataSync-6.0.0.0 (Build
20121022173845Z, Revision 11281)" authorizationType="Basic"
 requestContentType="appli-
cation/json" statusCode=200 etime=10.615 responseContentLength=0
```

Configuring HTTP Connection Handlers

HTTP connection handlers are responsible for managing the communication with HTTP clients and invoking servlets to process requests from those clients. They can also be used to host web applications on the server. Each HTTP connection handler must be configured with one or more HTTP servlet extensions and zero or more HTTP operation log publishers.

If the HTTP Connection Handler cannot be started (for example, if its associated HTTP Servlet Extension fails to initialize), then this will not prevent the entire Directory Server from starting. The Directory Server's `start-server` tool will output any errors to the error log. This allows the Directory Server to continue serving LDAP requests even with a bad servlet extension.

The configuration properties available for use with an HTTP connection handler include:

- **listen-address**. Specifies the address on which the connection handler will listen for requests from clients. If not specified, then requests will be accepted on all addresses bound to the system.
- **listen-port**. Specifies the port on which the connection handler will listen for requests from clients. Required.
- **use-ssl**. Indicates whether the connection handler will use SSL/TLS to secure communications with clients (whether it uses HTTPS rather than HTTP). If SSL is enabled, then `key-manager-provider` and `trust-manager-provider` values must also be specified.
- **http-servlet-extension**. Specifies the set of servlet extensions that will be enabled for use with the connection handler. You can have multiple HTTP connection handlers (listening on different address/port combinations) with identical or different sets of servlet extensions. At least one servlet extension must be configured.
- **http-operation-log-publisher**. Specifies the set of HTTP operation log publishers that should be used with the connection handler. By default, no HTTP operation log publishers will be used.
- **key-manager-provider**. Specifies the key manager provider that will be used to obtain the certificate presented to clients if SSL is enabled.
- **trust-manager-provider**. Specifies the trust manager provider that will be used to determine whether to accept any client certificates presented to the server.
- **num-request-handlers**. Specifies the number of threads that should be used to process requests from HTTP clients. These threads are separate from the worker threads used to process other kinds of requests. The default

value of zero means the number of threads will be automatically selected based on the number of CPUs available to the JVM.

- **web-application-extension**. Specifies the Web applications to be hosted by the server.

To Configure an HTTP Connection Handler

An HTTP connection handler has two dependent configuration objects: one or more HTTP servlet extensions and optionally, an HTTP log publisher. The HTTP servlet extension and log publisher must be configured prior to configuring the HTTP connection handler. The log publisher is optional but in most cases, you want to configure one or more logs to troubleshoot any issues with your HTTP connection.

1. The first step is to configure your HTTP servlet extensions. The following example uses the ExampleHTTPServletExtension in the Server SDK.

```
$ bin/dsconfig create-http-servlet-extension \
  --extension-name "Hello World Servlet" \
  --type third-party \
  --set "extension-
class:com.unboundid.directory.sdk.examples.ExampleHTTPServletExtension" \
  --set "extension-argument:path=/" \
  --set "extension-argument:name=example-servlet"
```

2. Next, configure one or more HTTP log publishers. The following example configures two log publishers: one for common access; the other, detailed access. Both log publishers use the default configuration settings for log rotation and retention.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Common Access Logger" \
  --type common-log-file-http-operation \
  --set enabled:true \
  --set log-file:logs/http-common-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"

$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Detailed Access Logger" \
  --type detailed-http-operation \
  --set enabled:true \
  --set log-file:logs/http-detailed-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

3. Configure the HTTP connection handler by specifying the HTTP servlet extension and log publishers. Note that some configuration properties can be later updated on the fly while others, like `listen-port`, require that the HTTP Connection Handler be disabled, then re-enabled for the change to take effect.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "Hello World HTTP Connection Handler" \
  --type http \
  --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set "http-servlet-extension:Hello World Servlet" \
  --set "http-operation-log-publisher:HTTP Common Access Logger" \
  --set "http-operation-log-publisher:HTTP Detailed Access Logger" \
  --set "key-manager-provider:JKS" \
  --set "trust-manager-provider:JKS"
```

4. By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to TRUE by default. You can monitor the connection handler using the `ldapsearch` tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

To Configure an HTTP Connection Handler for Web Applications

1. Create the Web application servlet extension.

```
$ bin/dsconfig create-web-application-extension \
  --extension-name "Hello Web Application" \
  --set "base-context-path:/hello-app" \
  --set "document-root-directory:/opt/hello-web-app"
```

2. By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to TRUE by default. You can monitor the connection handler using the `ldapsearch` tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

Domain Name Service (DNS) Caching

If needed, two global configuration properties can be used to control the caching of hostname-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system. Use the `dsconfig` tool to configure these properties.

- **network-address-cache-ttl** – Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a hostname-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.
- **network-address-outage-cache-enabled** – Caches hostname-to-IP address results in the event of a DNS outage. This is set to true by default, meaning name resolution results are cached. Unexpected service interruptions may occur during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

IP Address Reverse Name Lookups

PingData servers do not explicitly perform numeric IP address-to-hostname lookups. However, address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing may trigger implicit reverse name lookups. For more information about how address masks are configured in the server, review the following information for each server:

- ACI dns: bind rules under *Managing Access Control* (Directory Server and Directory Proxy Server)
- `ds-auth-allowed-address`: *Adding Operational Attributes that Restrict Authentication* (Directory Server)
- Connection Criteria: *Restricting Server Access Based on Client IP Address* (Directory Server and Directory Proxy Server)
- Connection Handlers: restrict server access using Connection Handlers (Configuration Reference Guide for all servers)

Configuring Traffic Through a Load Balancer

If a PingData server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, then it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request. If the actual client's IP address must be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and PingData server. For PingData servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP), and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

On the load balancer, configure settings to provide the `X-Forwarded-*` information, such as `X-Forwarded-Host:`. See the product documentation for the device type.

Working with the Referential Integrity Plug-in

Referential integrity is a plug-in mechanism that maintains the DN references between an entry and a group member attribute. For example, if you have a group entry consisting of member attributes specifying the DNs of printers, you can enable the referential integrity plug-in to ensure that the group entry is automatically removed if a printer entry is removed from the Directory Server.

The Referential Integrity Plug-in is disabled by default. When enabled, the plug-in performs integrity updates on the specified attributes (for example, member or uniquemember) after a delete, modify DN, or a rename (i.e., subordinate modifyDN) operation is logged to the `logs/ referint` file. If an entry is deleted, the plug-in checks the log file and makes the corresponding change to the associated group entry.

Three important points about the Referential Integrity Plug-in:

- All specified attributes that are configured for Referential Integrity must be indexed.
- On replicated servers, the Referential Integrity Plug-in configuration is not propagated to other replicas; therefore, you must manually enable the plug-in on each replica.
- The plug-in settings must also be identical on all machines.
- Subtree delete operations are not allowed if the referential integrity plugin is enabled and configured to operate in synchronous mode. It must be configured to operate in asynchronous mode (by specifying a nonzero update interval) if subtree delete operations will be performed.

To Enable the Referential Integrity Plug-in

1. Determine the attributes needed for your system. By default, the `member` and the `uniquemember` attributes are set for the plug-in.
2. Run the `dsconfig` tool to enable the Referential Integrity Plug-in.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

Working with the Unique Attribute Plug-in

The Unique Attribute plug-in is used to enforce uniqueness constraints on the values of one or more attributes across a portion of the Directory Server. The plug-in checks for uniqueness prior to an add, modify, or modify DN request and will instruct the server to reject the request if a constraint violation is found.

The plug-in is disabled by default as it can affect performance in heavy write load environments. Once the plug-in is enabled, it does not check for attribute uniqueness on existing entries, but only on new ADD, MODIFY, or MODDN operations. However, administrators can use the `identify-unique-attribute-conflicts` tool to ensure that no such conflicts exist in the data.

Important: All attributes for which uniqueness should be enforced should be indexed for equality in all backends. The LDAP SDK uniqueness request control can be used for enforcing uniqueness on a per-request basis. See the LDAP SDK documentation and the `com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl` class for using the control. See the ASN.1 specification to implement support for it in other APIs.

Attribute uniqueness can be enforced in replicated environments in which each replica contains the complete set of data for which to provide uniqueness, regardless of whether clients communicate directly with the server or interact with it through a Directory Proxy Server. In such environments, all servers should have identical uniqueness configurations. Note that it is not possible to *completely* prevent conflicts that arise from simultaneous writes on separate replicas. However, such conflicts will be detected after the changes have been replicated and will trigger administrative alert notifications.

For proxied environments that do not have the complete set of data on all servers (e.g., environments that use entry balancing or that store different portions of the DIT on different servers), you can implement the Global Uniqueness Attribute Plug-in on the Directory Proxy Server, instead of enabling the attribute uniqueness plug-in on the Directory Server. For more information, see the PingDirectoryProxy Server Administration Guide.

To Enable the Unique Attribute Plug-in

1. Determine which attributes must be unique in your data.
2. Run the `dsconfig` tool to enable the plug-in. By default, the plug-in type property is set to `postsynchronizationadd`, `postsynchronizationmodify`, `postsynchronizationmodifydn`, `preoperationadd`, `preoperationmodify`, and `preopertionmodifydn`. If you want to set one plug-in type, use the --set plugin-type:<operation-type> option. For example, use --set plugin-type:preoperationadd with the following command if you only want to check for attribute uniqueness prior to ADD operation.

```
$ bin/dsconfig set-plugin-prop --plugin-name "UID Unique Attribute" \
  --set enabled:true
```

Working with the Purge Expired Data Plug-in

The Purge Expired Data plug-in is used to delete expired entries or attribute values, and cleanup expired PingFederate Persistent Access Grants. When the plug-in is enabled, a background thread in the plugin periodically searches for and purges expired data. It is recommended that the Purge Expired Data plug-in be enabled on multiple servers in a topology. One server can be configured to delete data, while others can be searching for expired data.

The Purge Expired Data plug-in is created and configured with the `dsconfig` tool. Configuration options include the base DN and filter, the items to be purged, how to identify expired data, and the frequency for polling and purging. The search for expired data must be indexed. An alarm is raised if the server purging data falls behind the confiured `max-updates-per-second`. Monitoring information is available in the Admin Console, or `cn=monitor`.

To Configure the Purge Expired Data Plug-in for Expired Entries

This example deletes all unverified account entries that have not been accessed in the past eight weeks. They could be accounts that potential customers started to create through an application's registration process, but then did not complete. The phone number or email address that was provided during registration was not verified, and should be allowed to be used by another account. The server can track the last access time automatically in the `ds-last-access-time` attribute by enabling the Last Access Time plug-in.

1. If necessary, enable the Last Access Time plug-in:

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Last Access Time" \
  --set enabled:true
```

2. The Purge Expired Data Plug-in requires the date attribute that is used to determine expiration to be indexed for ordering. An index on that attribute must be created:

```
$ bin/dsconfig create-local-db-index \
  --backend-name userRoot \
  --index-name ds-last-access-time \
  --set index-type:ordering
```

3. If there is data present in the directory, rebuild the index:

```
$ bin/rebuild-index \
  --baseDN dc=example,dc=com \
  --index ds-last-access-time
```

4. Create the plug-in that purges account entries `objectclass=account` that are not verified `verified=false` after eight weeks of inactivity:

```
$ bin/dsconfig create-plugin \
  --plugin-name "Purge Old Unvalidated Accounts" \
  --type purge-expired-data \
  --set enabled:true \
  --set datetime-attribute:ds-last-access-time \
  --set "expiration-offset:8 w" \
  --set "filter:(&(objectClass=account)(verified=false))"
```

To Configure the Purge Expired Data Plug-in for Expired Attribute Values

The Purge Expired Data plug-in can also be used to delete values of an attribute that have expired. For example, an application may track information about an employee’s session, but the session should expire after 24 hours. There may be multiple active sessions tracked across different devices, with session information that looks like this:

```
sessionInfo: { “sessionId” : “E85FAC04E331FFCA55549B10B7C7A4FA”,
"ipAddress": "10.0.0.00", "userAgent": "Mozilla/5.0 (iPad; U; CPU OS 3_2
 like Mac OS X; en-us)
AppleWebKit/531.21.10 (KHTML, like Gecko) Version/4.0.4 Mobile/7B367
 Safari/531.21.10",
“creationTime” : “2018-03-31T13:10:15Z” }
```

In this example, the LDAP attribute is `sessioninfo`, and the JSON field that stores the time stamp is `creationTime`. These will be used to configure the Purge Expired Data plug-in.

To purge the JSON attribute values after 24 hours, rather than the entire session entry, the plug-in can be created with the following steps.

1. Create an index on the `creationTime` field of the `sessioninfo` attribute:

```
$ bin/dsconfig create-json-attribute-constraints \
  --attribute-type sessioninfo \
  --set enabled:true
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type sessioninfo \
  --json-field creationTime \
  --set index-values:true \
  --set value-type:string
```

2. Create and enable the plug-in:

```
$ bin/dsconfig create-plugin \
  --plugin-name "Purge Old Session Data Plugin" \
  --type purge-expired-data \
  --set enabled:true \
  --set "custom-datetime-format:yyyy-MM-dd'T'HH:mm:ss'Z'" \
  --set datetime-attribute:sessioninfo \
  --set datetime-format:custom \
  --set datetime-json-field:creationTime \
  --set "expiration-offset:1 d" \
  --set purge-behavior:delete-json-attribute-values
```

Configuring Uniqueness Across Attribute Sets

Attribute uniqueness can be configured across a set of attributes using the `multiple-attribute-behavior` property. The `multiple-attribute-behavior` property can take the following values:

- **unique-within-each-attribute** - If multiple attributes are specified, then uniqueness will be enforced for all values of each attribute, but the same value may appear in different attributes (in the same entry or in different entries). For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-within-each-attribute"` and add:

  An entry with a `telephoneNumber` value that matches the `telephoneNumber` attribute in the existing entry, then the add request will fail.

  An entry with a `mobile` value that matches the `mobile` attribute in the existing entry, then that too will fail.

  An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., 123-456-7893) but differ from the values in the existing entry, then the add request will succeed.
- **unique-across-all-attributes-including-in-same-entry** - If multiple attributes are specified, then uniqueness will be enforced across all of those attributes, so that if a value appears in one of those attributes, that value may not be present in any other of the listed attributes in the same entry, nor in any of the listed attributes in other entries. For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-across-all-attributes-including-in-same-entry"` and add:

  An entry with a `telephoneNumber` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then the add request will fail.

  An entry with a `mobile` value that matches the `mobile` attribute in an existing entry, then that too will fail.

  An entry with a `mobile` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then that will fail.

  An entry with a `telephoneNumber` value (e.g., 123-456-7891) that matches the `mobile` attribute in an existing entry, then that too will fail.

  An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., 123-456-7893) but differ from the values in an existing entry, then the add request will fail.
- **unique-across-all-attributes-except-in-same-entry** - If multiple attributes are specified, then uniqueness will be enforced across all of those attributes, so that if a value appears in one of those attributes, that value may not be present in any of the listed attributes in other entries. However, the same value may appear in multiple attributes in the same entry. For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-across-all-attributes-except-in-same-entry"` and add:

  An entry with a `telephoneNumber` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then the add request will fail.

  An entry with a `mobile` value that matches the `mobile` attribute in the existing entry, then that too will fail.

  An entry with a `mobile` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then that will fail.

  An entry with a `telephoneNumber` value (e.g., 123-456-7891) that matches the `mobile` attribute in an existing entry, then that will fail.

  An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., 123-456-7893) but differ from the values in an existing entry, then the add request will succeed.

To Enable Uniqueness Across Attribute Sets

- Use `dsconfig` to configure the UID Unique Attribute plug-in to apply across multiple attributes. The `multiple-attribute-behavior` property is set to "`unique-within-each-attribute`", which indicates that uniqueness will be enforced for all values of each attribute (e.g., `telephoneNumber=123-456-7890` and `mobile=123-456-7891`), but the same value (e.g., either 123-456-7890 or 123-456-7891) may appear in different attributes in the same entry or in different entries.

```
$ dsconfig create-plugin \
  --plug-in "Unique telephoneNumber and mobile" \
  --type "unique-attribute" \
  --set "enabled:true" \
  --set "type:telephoneNumber" \
  --set "type:mobile" \
  --set "multiple-attribute-behavior:unique-within-each-attribute" \
  --no-prompt
```

Working with the Last Access Time Plug-In

The Last Access Time plug-in is used to record the timestamp of the last activity targeting an entry. The plug-in updates the `ds-last-access-time` attribute of the entry when it is accessed by an add, bind, compare, modify, modify DN, or search operation.

The plug-in can be used with the Directory Server Uncached Attribute Criteria, or any application that needs to determine how recently an entry has been accessed. The plug-in also enables defining request criteria to limit the scope of tracking the last access time. The `max-search-result-entries-to-update` property also prevents mass updates of `ds-last-access-time` when searches contain many results, but may not reflect end-user access. Consider the following when using this plug-in:

- The plugin should be enabled on all servers that have the same configuration.
- An updated `ds-last-access-time` attribute value is replicated like any other change to an entry.
- The `ds-last-access-time` attribute is not returned from a search, unless included in the attributes list explicitly, or given the "+" specification for operational attributes.
- The `ds-last-access-time` value format is `yyyyMMddHHmmss.SSS'Z'`, which provides millisecond-level accuracy, such as `20131207144135.821Z`.
- The `ds-last-access-time` attribute can be indexed with a local database index. The ordering index type is the most relevant, but may require a higher index entry limit (default is 4000) to accommodate searches for entries that have not been accessed in a long period of time. The ordering index type, with a short time range or high index entry limit, will result in indexed search results for requests such as `(ds-last-access-time>=20131207144135.821Z)`.

☞ **Important:** Deployments prior to version 4.5 using the last access time plug-in should disable the plug-in before upgrading, and then re-enable the plug-in once the update is complete. If servers are running different versions, the `last-access-time` updates may occur with a higher frequency than intended.

Working with the Pass Through Authentication Plug-In

The Pass Through Authentication plugin is used to delegate bind operations to remote LDAP servers by forwarding simple bind requests to an external LDAP server, including Active Directory. The plugin can be configured to attempt a local bind, set or update a local password, and bypass local password policies to ensure remote passwords are migrated.

Consider the following when using this plugin:

- The plugin should be enabled on all servers that have the same configuration.
- Remote servers accepting a forwarded bind request may require connection security, such as a secure StartTLS or LDAPS TLS connection.
- Carefully consider how password changes and password resets are handled. Updating a password in the Directory Server may result in divergent passwords between the local and remote server. The Data Sync Server can be used to synchronize passwords between servers, if needed.
- The plugin only updates local passwords if the forwarded simple bind is successful. Expired passwords on a remote server may return an invalid credentials error causing the overall bind operation to fail.

- Multiple remote servers can be specified. The server-access-mode property determines if the servers are accessed in round-robin, failover-on-unavailable, or failover-on-any-failure modes. The default server access mode is round-robin.
- The update-local-password property indicates whether the local password value should be updated to the value used in the bind request, in the event that the local bind fails but the forwarded bind succeeds. A local entry must previously exist in order to update passwords.
- The allow-lax-pass-through-authentication-passwords property indicates whether updates to the local password value should accept passwords that do not meet local password policy requirements.
- The connection-criteria property specifies a set of connection criteria that must match the client associated with the local bind request for the bind to be forwarded to the remote server.
- The request-criteria property specifies a set of request criteria that must match the local bind request or a local target entry for the bind to be forwarded to the remote server.
- The dn-map property specifies one or more DN mappings that can be used to transform bind DNs before attempting to forward the bind to remote servers.
- The search-base-dn property is used when searching for a remote user entry using a filter constructed from the pattern defined in the search-filter-pattern property. It is not possible to configure both a DN map and search filter pattern. If neither a DN map nor a search filter pattern is defined, then user entries are expected to have the same DN in the local server and the remote servers.

Supporting Unindexed Search Requests

By default, the Directory Server denies all unindexed search requests, except for those issued by the bind DNs that have the unindexed-search privilege. This default behavior keeps the server from tying up worker threads on time-consuming, unindexed searches. However, you can turn off the enforcement of the unindexed-search privilege to allow any client to perform an unindexed search. To do this, set the disabled-privilege global configuration property to unindexed-search as follows:

```
$ bin/dsconfig set-global-configuration-prop \
  --set disabled-privilege:unindexed-search
```

If you choose to allow unindexed searches, you may want to cap the maximum number of concurrent unindexed search requests using the maximum-concurrent-unindexed-searches global configuration property. You configure this property using dsconfig as follows:

```
$ bin/dsconfig set-global-configuration-prop \
  --set maximum-concurrent-unindexed-searches:2
```

You can limit unindexed search privileges for particular clients using the allow-unindexedsearches property of the Client Connection Policy. For more information about configuring Client Connection Policies, see “Configuring Client Connection Policies”.

Sun/Oracle Compatibility

For companies that are migrating from a Sun/Oracle server to the PingDirectory Server, the PingDirectory Server provides a dsconfig batch file, sun-ds-compatibility.dsconfig, which describes the various components that can be configured to make the server exhibit behavior closer to a Sun/Oracle configuration.

Administrators can use the sun-ds-compatibility.dsconfig batch file to apply the Directory Server’s configuration with the necessary dsconfig commands. Simply uncomment the example commands listed in the file, and then run the dsconfig command specifying the batch file. Note that this batch file is not comprehensive and must be used together with the migrate-sun-ds-config tool, located in the bin folder (or bat folder for Windows systems) during the migration process. Both the tool and the batch file overlap in some areas but provide good initial migration support from the Sun/Oracle server to a Ping Identity server.

Another useful tool is the `migrate-ldap-schema` tool in the `bin` folder (or `bat` folder for Windows systems), which migrates schema information from an existing LDAP server onto this instance. All attribute type and object class definitions that are contained in the source LDAP server will be added to the targeted instance or written to a schema file.

To Configure the Directory Server for Sun/Oracle Compatibility

Content-Reference to:

1. From the Directory Server server root directory, open the `sun-ds-compatibility.dsconfig` file in the `docs` folder. You can use a text editor to view the file.
2. Read the file completely.
3. Apply any changes to the file by removing the comment symbol at any `dsconfig` command example, and then applying the `dsconfig` command specifying the batch file.

```
$ bin/dsconfig --no-prompt --bindDN "cn=Directory Manager" \
  --bindPassword "password" --batch-file /path/to/dsconfig/file
```

4. Run the `migrate-ldap-schema` tool to move the schema definitions on the source server to the destination Ping Identity server.

```
$ bin/migrate-ldap-schema
```

5. Next, run the `migrate-sun-ds-config` tool to see what differences exist in the Ping Identity configuration versus the Sun/Oracle configuration. On the PingDirectory Server, run the following command:

```
$ bin/migrate-sun-ds-config
```

6. Test the server instance for further settings that may not have been set with the batch file, the `migrate-ldap-schema` tool, or the `migrate-sun-ds-config` tool.
7. If you notice continued variances in your configuration, contact your authorized support provider.

Chapter

7

Configuring Soft Deletes

Topics:

The PingDirectory Server (version 3.2.4 or later) supports a *soft-delete* feature that preserves a deleted entry's attribute and uniqueness characteristics to allow it to be undeleted or permanently removed at a later date.

This chapter introduces the following topics:

About Soft Deletes

The standard implementation of an LDAP server allows for adding, renaming, modifying, searching, comparing, and deleting one or more entries. The DELETE operation is, by specification, a destructive operation that permanently removes an entry and its attributes in a Directory Information Tree (DIT) but records the changes in access, and optionally, audit and change logs. During the DELETE operation, any associations such as references and group memberships are severed to reflect the entry that is removed. Meta attributes like operational attributes, which may be unique to an entry like `entryUUID`, will be lost or be different if the same entry is re-added to the Directory Server.

There are cases, however, where a company may want to preserve their deleted entries to allow for possible undeletion at a later date. For example, a company may want to retain account and subscriber entries for their users (e.g., customers, employees, or partners) who leave but later rejoin. Artifacts that a user creates such as account histories, web pages, notes, may be tracked and recovered while a user is deleted or when the user returns as an active customer.

To facilitate this use case, the Directory Server supports a feature called *soft deletes*, which preserves a deleted entry's attributes and entry uniqueness characteristics to allow the entry to be undeleted or permanently removed at a later date. A delete request may result in a soft delete either by the client explicitly requesting a soft delete or by the request matching criteria defined in an active soft delete policy. The soft-deleted entries are renamed by prefixing an `entryUUID` operational attribute to the DN and adding an auxiliary object class, `ds-soft-delete-entry`, to the entry, which saves the entry in a hidden state. All active references and group memberships are then removed. Once in this hidden state, soft-deleted entries are inaccessible to clients under normal operating conditions. Only clients with the `soft-delete-read` privilege will be allowed to interact with soft-deleted entries.

To allow soft deletes, the Directory Server's attribute uniqueness function has been relaxed to allow for the co-existence of a soft-deleted entry and an active entry with identical naming attributes, such as `uid`. For example, if a user John Smith was soft deleted but a different John Smith was added to the user accounts system, both entries could reside in the DIT without conflict: one in a soft-deleted state; the other, in an active state. The Directory Server extends this capability further by allowing multiple users with the same DN, who would normally conflict if active, to reside in the soft-deleted state.

Soft-deleted entries can be restored with an Undelete operation. However, the same uniqueness constraints that apply when adding a new user to the Directory Server are enforced when a soft-deleted entry is undeleted. Returning to the previous example, John Smith was soft deleted but a different John Smith with the same `uid` as the original John Smith was added later to the system. If the original John Smith was undeleted from its soft-deleted state, it would result in a conflict with the active John Smith entry. Administrators will need to modify the DN of the soft-deleted entry to avoid such conflicts.

Administrators can permanently remove a soft-deleted entry by performing a regular DELETE operation on it. This operation, called a *hard delete*, permanently removes a soft-deleted entry from the server. Also, note that you can also permanently remove a regular non-soft-deleted entry using a hard delete. This is useful when the server is configured with a soft-delete policy that would otherwise turn a regular delete request into a soft delete.

The Directory Server provides tool arguments that can use the Soft Delete Request Control, a Hard Delete Request Control, and other controls necessary to process these operations. Procedures to show how to use these options are presented later in this section.

For replicated topologies, when a participating directory server soft deletes an entry, it notifies the other replicas in the topology to soft delete the same entry on its respective machine. The changelog backend also records these entries by annotating them using an attribute that indicates its soft-deleted state. Modification and hard deletes of soft-deleted entries are not recorded by default in the changelog but can be enabled in the server. For maximum compatibility, it is highly recommended that all servers in the replication cluster support Soft Deletes and have identical Soft Delete configurations.

General Tips on Soft Deletes

There are some general tips about soft deletes that administrators should be aware of:

- **LDAP SDK and Server SDK**. The LDAP SDK and Server SDK both fully supports soft-deletes.
- **Possible Performance Impact for Searching Regular Entries and Soft-Deleted Entries**. There is little performance difference between retrieving a regular entry and a soft-deleted entry, respectively. However, there may be a performance impact when a search operation has to match criteria (such as, `uid=john.smith`) for both active entries and soft-deleted entries. For example, if there is one active `uid=john.smith` entry and two soft-deleted `uid=john.smith` entries, it may take the server a little more time to retrieve and try to match the criteria before it can return the results.
- **Soft Delete for Uncached Attributes and Entries**. The soft delete feature fully supports uncached attributes and uncached entries. See the section on *Uncached Attributes and Entries* for more information about the feature.
- **Soft Delete for Leaf Nodes Only**. Soft-deletion of any parent entry is not allowed. Likewise, soft-deleted entries that have soft-deleted sub-entries are not allowed.
- **Attempting to Soft-Delete a Soft-Deleted Entry Fails**. There are two available state options for soft-deletes: administrators can permanently delete a soft-deleted entry or undelete the entry. An administrator cannot soft-delete an already soft-deleted entry, which returns an `UNWILLING_TO_PERFORM` result code.
- **Soft-Deleted Users Have No Privileges**. Soft-deleted users do not have the ability to bind to the Directory Server or have authentication access. They cannot change their passwords and cannot undelete themselves. Also, soft-deleted entries cannot be used as an authorization identity using the proxied authorization or immediate client control. It is important to note that the soft-delete process does not destroy privilege assignment. If a soft-deleted entry is undeleted, the restored entry will retain the same privileges it originally had before being soft deleted. (One possible exception to this are those privileges assigned by virtual attributes that no longer match the newly-undeleted entry; those entries do not retain their original privileges.)
- **Soft-Deleted Entries Not Accessible by Other Means**. Soft-deleted entries may not be accessible from alternate access methods like SCIM.
- **Soft-Deleted Entries Can Be Modified but Not Renamed**. Administrators can search for all soft-deleted entries and the original source entry attributes can be updated as long as the administrators has modify privileges and access to the `soft-delete-read` privilege. Any attempt to rename a soft-deleted entry using a MODIFY DN operation will result in an `UNWILLING_TO_PERFORM` result code.
- **Replication**. Replication will have access to the LDAP operations with Soft Delete controls. These operations are transmitted, processed, and replayed as high-level requests, which are re-played on remote replicated servers. The replication conflict-resolution mechanism handles soft-deleted entries like any regular entries. For example, if a soft delete is executed independently on two servers then replicated, this results in a replication conflict. For maximum compatibility, it is highly recommended that all servers in a replication cluster support Soft Deletes and have identical Soft Delete configuration.
- **Transactions**. Soft-deletes are supported in transactions. The processing workflow uses the transactions mechanism and maintains the context information necessary to rollback failures to soft delete or undelete.
- **Soft-Deletes Through the Directory Proxy Server**. There is no special configuration steps to configure Soft Deletes on the Directory Proxy Server. The soft-deleted entry is routed directly to the underlying Directory Server. There is one exception: in an entry-balancing deployment, the Directory Proxy Server is responsible for routing the soft-deleted entry to the Directory Server containing the originally soft-deleted item. Also, as with standard entry-balanced deployments, it is not possible to undelete (using MODDN) an entry to a different Directory Server.
- **Export-LDIF**. The default behavior is to include soft-deleted entries as part of the `export-ldif` operation. If soft-deleted entries are to be excluded from export, administrators can use the --excludeSoftDeleteEntries option to filter out the entries.
- **Proxied Authorization**. The Soft Delete feature can be used with users who have proxied authorization privileges.
- **Ignored by Data Sync Server Sync Pipes**. For customers using the PingDataSync Server, soft-deleted entries are not synchronized by the server. Modifications or deletes of a soft-deleted entry are ignored by the Data Sync Server, and do not appear in the changelog by default. An actual soft delete operation appears to the changelog as a regular DELETE, and an actual undelete operation appears in the changelog as a regular ADD.
- **Referential Integrity Plugin Does Not Restore Membership**. References to a deleted DN are not restored by the referential integrity plugin upon undeletion of a soft-deleted entry. For example, if you have referential integrity enabled and you soft-delete a DN that is a member of a static group, the referential integrity plugin will remove

this DN from the group's list of members. When you undelete the soft-deleted entry, the plugin will not add the entry back to the group.

- **Criteria-Selected or Explicitly-Requested Purging of Soft Deletes**. The Soft Delete Policy configuration supports two new properties, `soft-delete-retention-time` and `soft-delete-retain-number-of-entries` that performs purging of soft deleted entries. See the section on *Configuring Soft-Delete Automatic Purging*.
- **Assigning Access to Controls to Non-Root Users Administrators**. By default, the root user account (e.g., `cn=Directory Manager`) has access to all of the controls needed to run the Soft Delete operations. For non-root users, you must grant access to these Soft Delete controls using access control rules. An example is shown in step 1 of the section, *To Configure Soft Deletes as a Global Configuration*. The following Soft Delete Controls are available to non-root users:

  Soft Delete Request Control. Allows the user to perform a soft delete operation. The OID for the control is 1.3.6.1.4.1.30221.2.5.20.

  Soft Delete Response Control. Allows the server to hold the DN of the soft-deleted entry that results from a soft delete request. The OID for the control is 1.3.6.1.4.1.30221.2.5.21.

  Hard Delete Request Control. Allows the user to run a hard delete operation on the entry, regardless if it is a regular or soft-deleted entry. The OID for the control is 1.3.6.1.4.1.30221.2.5.22.

  Undelete Request Control. Allows the user to undelete a soft-deleted entry using an ADD request. The OID for this control is 1.3.6.1.4.1.30221.2.5.23.

  Soft-Deleted Entry Access Request Control. Allows the user to search for any soft-deleted entries. The OID for this control is 1.3.6.1.4.1.30221.2.5.24. Note that a bind DN with the `stream-values` privilege can perform operations that can reveal soft-deleted entries, even if that bind DN does not have permission to use the Soft-Deleted Entry Access Request Control. For example, if a user can successfully run `dump-dns` or `ldap-diff`, then that user can get a list of soft-deleted entry DNs or soft-deleted entry contents via the output of one of those tools.

Configuring Soft Deletes on the Server

Soft deletes are configured on the Directory Server in several ways:

- **Using a Soft-Delete Policy and Global Configuration Property**. You can configure soft deletes by creating a soft-delete policy and a global configuration property. The soft-delete policy enables the feature on the server, while the global configuration property sets the controls used for the soft-delete requests. To enter a soft delete request, the `ldapmodify` or `ldapdelete` command requires the --useSoftDelete option. A delete request that does not have the --useSoftDelete option is treated as a hard delete, which permanently removes the entry.
- **Using Connection Criteria**. You can configure soft deletes by defining connection criteria within a client connection policy. Any clients that meet the criteria will have their deletes processed as soft deletes.
- **Using Request Criteria**. You can configure soft deletes by defining request criteria within a client connection policy. Any client requests that meet the criteria will have their deletes processed as soft deletes. Note that you can define both connection criteria and request criteria. Both criteria are exclusive and can exist within a soft-delete policy. In this case, the connection criteria is processed first, then the request criteria.

Configuring Soft Deletes as a Global Configuration

You can configure the soft delete feature by creating a soft-delete policy and then setting the configuration property on the server. The presence of the soft-delete policy enables the feature on the server and allows the global configuration property to send the necessary soft-delete requests.

This configuration setting requires that the --useSoftDelete option be used together with the `ldapmodify` or `ldapdelete` commands to send the delete using the Soft Delete Request Control. Without the --useSoftDelete option, any delete will be processed as a hard delete.

To Configure Soft Deletes as a Global Configuration

1. Create a non-root user admin account, such as `uid=admin,dc=example,dc=com`. See steps 1 and 3 in the Configuring Administrator Accounts section for more information. The following LDIF adds ACI's to the

`uid=admin,dc=example,dc=com` entry to allow access to user and operational attributes as well as access to the controls necessary to carry out the Soft Delete operations. The LDIF also adds the `soft-delete-read` privilege, which allows the user to access soft-deleted entries. Add the LDIF file using the `ldapmodify` tool.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*||+")
      (version 3.0; acl "Allow admins to read and write all user and
 operational
       attributes";
       allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.21")
      (version 3.0; acl "Allow admins to use the Soft Delete Request Control
 and
       Soft Delete Response Control";
       allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.22")
      (version 3.0; acl "Allow admins to use the Hard Delete Request
 Control";
       allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.23")
      (version 3.0; acl "Allow admins to use the Undelete Request Control";

       allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.24")
      (version 3.0; acl "Allow admins to use the Soft-Deleted Entry Access
 Request
       Control";
       allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)

$ bin/ldapmodify --fileName add-admin-aci.ldif
```

2. Create another LDIF file and assign the `soft-delete-read` privilege to the authorized user. Add the file to the server using the `ldapmodify` tool. You'll need to include bind credentials with the `ldapmodify` command.

```
dn: uid=admin,dc=example,dc=com
changeType: modify
add: ds-privilege-name
ds-privilege-name: soft-delete-read

$ bin/ldapmodify --fileName add-admin-priv.ldif
```

3. Configure a soft delete policy using the `dsconfig` command. The soft delete configuration option requires a soft-delete policy, which effectively enables the feature on the server. This is a required step.

```
$ bin/dsconfig create-soft-delete-policy \
  --policy-name default-soft-delete-policy
```

4. Configure the soft delete as a global configuration property using the `dsconfig` command. The command sets up the soft-delete controls necessary to send them as a request. This is a required step.

```
$ bin/dsconfig set-global-configuration-prop \
  --set soft-delete-policy:default-soft-delete-policy
```

After a successful modification, the server issues a warning that soft deletes are not enabled and that administrative action may be needed to prune older soft-deleted entries if resources are limited. For more information on setting up soft deletes, see "Configuring Soft Deletes by Connection Criteria".

```
One or more configuration property changes require administrative action or
confirmation/notification. Those properties include:
 * soft-delete-policy:  Presently the server does not have an automatic
 purging
   capability for soft deleted entries. The administrator must
   periodically monitor and determine if too many soft deleted entries are
   consuming resources and must be pruned.

The Global Configuration was modified successfully
```

5. Delete an entry to test the soft delete feature. You must use the --useSoftDelete (or the short form, -s) option with the `ldapdelete` or `ldapmodify` to send the delete using a Soft Delete Request Control. The control renames the entry, prefixes an `entryUUID` operational attribute to the DN, and hides the entry by adding an auxiliary object class, `ds-soft-delete-entry`. If you do not include the --useSoftDelete option, the server processes the delete as a hard delete, which permanently removes the entry unless soft deletes is enabled. Upon a successful soft delete operation, the server returns the DN of the newly soft-deleted entry.

```
$ bin/ldapdelete --useSoftDelete uid=user.1,ou=People,dc=example,dc=com
```

```
Processing DELETE request for uid=user.1,ou=People,dc=example,dc=com
DELETE operation successful for DN uid=user.1,ou=People,dc=example,dc=com

Soft-deleted entry DN:
entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4+uid=user.1,ou=People,dc=exam-
ple,dc=com
```

Note:

Once an entry has been soft deleted, it is possible for an LDAP client to create an entry with the same DN as the original user entry DN. For example, if you soft-delete an entry (e.g., `uid=user.3,ou=People,dc=example,dc=com`), you can create a new entry using the same DN as the original entry, `uid=user.3,ou=People,dc=example,dc=com`. If the soft-deleted entry is undeleted at a later date, then a conflict will occur.

Searching for Soft Deletes

Soft-deleted entries are excluded from normal LDAP searches because they represent "deleted" entries. The `ldapsearch` tool has been updated to support these types of searches. If you want the option to search for soft-deleted entries, there are three ways to do so:

- **Base-Level Search on a Soft-Deleted entry by DN**. Use `ldapsearch` and specify the base DN of the specific soft-deleted entry that you are searching for.
- **Filtered Search by `soft-delete-entry` object class**. To search for all soft-deleted entries, use `ldapsearch` with a filter on the `ds-soft-delete-entry` objectclass.
- **Soft-Delete-Entry-Access Control**. You can use the Soft Delete Entry Access Control with the LDAP search to return soft-deleted entries. The `ldapsearch` tool provides a shortcut option, `--includeSoftDeletedEntries`, that sends the control to the server for processing. The control allows for the following search possibilities:

  Return only soft-deleted entries.
  Return non-deleted entries along with soft-deleted entries.
  Return only soft-deleted entries in undeleted form.

To Run a Base-Level Search on a Soft-Deleted Entry

- Run `ldapsearch` using the base DN of the specified soft-deleted entry.

```
$ bin/ldapsearch \
  --baseDN entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com \
  --searchScope base "(objectClass=*)"
```

```
# Soft-deleted entry DN:
# entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-soft-delete-entry
postalAddress: Aartjan Aalders$59748 Willow Street$Green Bay, TN 66239
postalCode: 66239
description: This is the description for Aartjan Aalders.
uid: user.1
userPassword: {SSHA}RdBCwQ2kIw57LukRthjrFBS/oFylJARnmTnorA==
employeeNumber: 1
initials: AKA
givenName: Aartjan
pager: +1 197 025 3730
mobile: +1 890 430 9077
cn: Aartjan Aalders
sn: Aalders
telephoneNumber: +1 094 100 7524
street: 59748 Willow Street
homePhone: +1 332 432 4295
l: Green Bay
mail: user.3@maildomain.net
st: TN
```

To Run a Filtered Search by soft-delete-entry Object Class

- Run `ldapsearch` to retrieve all soft-deleted entries using the `ds-soft-delete-entry` object class. The following command retrieves all soft-deleted entries on the server

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  "(objectclass=ds-soft-delete-entry)"
```

To Run a Search using the Soft Delete Entry Access Control

The following examples use the --includeSoftDeleteEntries `{with-non-deleted-entries | without-non-deleted-entries | deleted-entries-in-undeleted-form}` option, which uses the Soft Delete Entry Access Control. You could also use the --control option with the Soft Delete Entry Access Control symbolic name, `softdeleteentryaccess`, or the `--control` option with the actual Soft Delete Entry Access Control OID, `1.3.6.1.4.1.30221.2.5.24`.

- **Return Only Soft-Deleted Entries**. Run `ldapsearch` using the --includeSoftDeletedEntries option with the value of `without-non-deleted-entries` to return only soft-deleted entries.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries without-non-deleted-entries \
  --searchScope sub "(objectclass=*)"
```

- **Return Non-Deleted Entries Along with Soft-Deleted Entries**. Run `ldapsearch` using the --includeSoftDeletedEntries option with the value of with-non-deleted-entries to return non-deleted entries along with soft-deleted entries.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries with-non-deleted-entries \
  --searchScope sub "(objectclass=*)"
```

- **Return Only Soft-Deleted Entries in Undeleted Form**. Run `ldapsearch` using the --includeSoftDeletedEntries option with the value of `deleted-entries-in-undeleted-form` to return only soft-deleted entries in undeleted form. Some applications require access to all entries in the server, including both active and soft-deleted entries. The following command returns all entries that were soft-deleted but presents it in a form that is similar to a regular entry with the soft-delete DN in comments. This regular entry format does not show the actual soft-deleted DN but displays it in an "undeleted" form even though it is not actually "undeleted". Also, the object class, `ds-soft-delete-entry`, is not displayed:.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries deleted-entries-in-undeleted-form \
  --searchScope sub "(ds-soft-delete-from-dn=*)"

# Soft-deleted entry DN:
# entryUUID=2b5511e2-7616-389b-ab0c-025c805ad32c
+uid=user.14,ou=People,dc=exam-
ple,dc=com
dn: uid=user.14,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Abdalla Abdou$78929 Hillcrest Street$Elmira, ME 93080
postalCode: 93080
description: This is the description for Abdalla Abdou.
uid: user.14
userPassword: {SSHA}7GkzWiMiU12m5m+xBV+ZsoX3gVacMcRtSwDTFg==
employeeNumber: 14
initials: AFA
givenName: Abdalla
pager: +1 307 591 4870
mobile: +1 401 069 1289
cn: Abdalla Abdou
sn: Abdou
telephoneNumber: +1 030 505 6190
street: 78929 Hillcrest Street
homePhone: +1 119 487 2328
l: Elmira
mail: user.14@maildomain.net
st: ME
```

Undeleting a Soft-Deleted Entry Using the Same RDN

To undelete a soft-deleted entry, use `ldapmodify` with the --allowUndelete option and target the specific soft-deleted entry that you want to restore. In an LDIF file or from the command line, specify the `dn:<target entry>` attribute, which is the DN that the entry will be undeleted *to* and the `ds-undelete-from-dn` attribute, which is the entry that will be undeleted *from*. An undelete requires the `add` changetype so that the entry can be re-added to the server.

To Undelete a Soft-Deleted Entry Using the Same RDN

- Use `ldapmodify` with the --allowUndelete option and target the specific soft-deleted entry that you want to restore. The --allowUndelete option sends the Soft Undelete Request Control to the server. The first DN is the entry to undelete to and the `ds-undelete-from-dn` is the soft-delete entry to undelete from.

```
$ bin/ldapmodify --allowUndelete
dn: uid=user.1,ou=People,dc=example,dc=com
changetype:add
ds-undelete-from-dn: entryUUID=4e9b7847-edcb-3791-b11b-
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Undeleting a Soft-Deleted Entry Using a New RDN

In some cases, the original RDN, `uid=user.1`, may have been allocated to a new user, which is allowed while the entry is in a soft-deleted state. To properly undelete this entry, you need to specify a new RDN value that the entry should be restored with. In this case, specifying the RDN of `uid=user.5` will undelete the original entry but with the new DN in the example below. In addition, the `uid` attribute on the entry will be updated with the new value of `user.5` as well. All other attributes of the users entry including the `entryUUID` will remain unchanged.

To Undelete a Soft-Deleted-Entry Using a New RDN

1. Use `ldapmodify` to undelete a soft-deleted entry that has an original RDN, `uid=user.1`, to a new RDN, `uid=user.5`. You will need to ensure that the DN that you are undeleting the entry to does not already exist as this will lead to an "entry already exists" error if you specify a DN that already exists in the Directory Server as a normal entry.

```
$ bin/ldapmodify --allowUndelete
dn: uid=user.5,ou=People,dc=example,dc=com
changetype:add
ds-undelete-from-dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

2. View the results using `ldapsearch`. You will notice the RDN and the `uid` attribute has changed.

```
dn: uid=user.5,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Aartjan Aalders$59748 Willow Street$Green Bay, TN 66239
postalCode: 66239
description: This is the description for Aartjan Aalders.
uid: user.5
userPassword: {SSHA}RdBCwQ2kIw57LukRthjrFBS/oFylJARnmTnorA==
employeeNumber: 1
initials: AKA
givenName: Aartjan
pager: +1 197 025 3730
mobile: +1 890 430 9077
cn: Aartjan Aalders
sn: Aalders
telephoneNumber: +1 094 100 7524
street: 59748 Willow Street
homePhone: +1 332 432 4295
l: Green Bay
mail: user.3@maildomain.net
st: TN
entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4
```

Modifying a Soft-Deleted Entry

You can modify a soft-deleted entry as you would a regular entry using the `ldapmodify` tool and remains hidden in its soft-deleted state after the change. The only restriction is that you cannot change the DN or run a MODDN operation on the soft-deleted entry.

To move a soft-deleted entry from one machine to another, use the `move-subtree` command and specify the DN of the soft-deleted entry. For more information, see *To Move an Entry from One Machine to Another*.

Note: To modify a soft-deleted entry, the user needs the `soft-delete-read` privilege to access the soft-deleted entry.

To Modify a Soft-Deleted Entry

- Soft-deleted entries can be modified like any regular entry. Use `ldapmodify` and specify the soft-deleted DN.

```
$ bin/ldapmodify
dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
changetype:modify
replace:telephoneNumber
telephoneNumber: +1 390 103 6918
# Processing MODIFY request for entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
# MODIFY operation successful for DN entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Hard Deleting a Soft-Deleted Entry

To permanently remove a soft-deleted entry from the server, you can run `ldapdelete` on the soft-deleted entry for soft-deleted entries. To hard delete a soft-deleted entry, use `ldapdelete` with the --useHardDelete option. The Hard Delete Request Control works with soft deletes. It mostly applies when soft delete policies are in place as a means to override soft deletes requests. If soft deletes are configured, running `ldapdelete` with the Hard Delete Request Control (i.e., using the option, --useHardDelete) guarantees any entry will be permanently deleted.

To Hard Delete a Soft-Deleted Entry (Global Configuration)

- Run `ldapdelete` on a soft-deleted entry to permanently remove it from the Directory Server. This example assumes that you configured soft deletes as a global configuration for requests.

```
$ bin/ldapdelete \
  entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com

Processing DELETE request for entryUUID=4e9b7847-edcb-3791-b11b-
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
DELETE operation successful for DN entryUUID=4e9b7847-edcb-3791-b11b-
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Note:

You cannot soft-delete an already soft-deleted entry. If you use the --useSoftDelete with the `ldapdelete` operation on a soft-deleted entry, an error message will be generated.

```
DELETE operation failed.
Result Code: 53 (Unwilling to Perform)
Diagnostic Message: DELETE operation failed.
```

To Hard Delete a Soft-Deleted Entry (Connection or Request Criteria)

- Run `ldapdelete` with the --useHardDelete option on a soft-deleted entry to permanently remove it from the server. This example assumes that you configured soft deletes using a connection or request criteria.

```
$ bin/ldapdelete --useHardDelete \
  entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Disabling Soft Deletes as a Global Configuration

To disable soft deletes on your Directory Server, simply reset the global configuration property and the remove the soft-delete policy. From that point, all deletes will be processed as hard deletes by default. Any use of the soft-deleted options with the LDAP tools results in an error. Any existing soft-deleted entries that are present after the global configuration is disabled will remain in the server as latent entries.

To Disable Soft Deletes as a Global Configuration

1. Run `dsconfig` to reset the global configuration property. Remember to include the LDAP bind parameters for your system.

```
$ bin/dsconfig set-global-configuration-prop --reset soft-delete-policy
```

2. Run `dsconfig` to delete the soft delete policy that you created. Remember to include the LDAP bind parameters for your system.

```
$ bin/dsconfig delete-soft-delete-policy --policy-name default-soft-delete-
policy
```

Configuring Soft Deletes by Connection Criteria

The Directory Server supports soft deletes where any delete operation is treated as a soft-delete request as long as the LDAP client meets the connection criteria. You can configure soft deletes by defining the connection criteria used in a client connection policy, and then configuring the soft delete connection criteria in the soft-delete policy.

To Enable Soft Deletes by Connection Criteria

1. Configure a soft-delete policy and global configuration as shown in “Configuring Soft Deletes as a Global Configuration”.
2. Create a simple connection criteria using `dsconfig` and name it "Internal Applications". The soft delete connection criteria is configured for a member of a Line of Business (LOB) Applications group connecting from the 10.8.1.0 network.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Internal Applications" \
  --type simple \
  --set included-client-address:10.8.1.0/8 \
  --set "all-included-user-group-dn:cn=LOB
 Applications,ou=Groups,dc=example,dc=com"
```

3. In the soft delete policy created in step 1, set the `auto-soft-delete-connection-criteria` property to the simple criteria created in the previous step.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set "auto-soft-delete-connection-criteria:Internal Applications"
```

To Disable Soft Deletes by Connection Criteria

- To disable soft deletes by connection criteria, simply reset the `auto-soft-delete-connection-criteria` property on the soft-delete policy.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset auto-soft-delete-connection-criteria
```

Configuring Soft Deletes by Request Criteria

Soft deletes can be configured using request criteria within a client connection policy. All delete requests that meet the request criteria is treated as a soft delete. The presence of a soft delete by connection criteria is exclusive of the soft delete by request criteria. Both can be present in a soft-delete policy.

To Enable Soft Deletes by Request Criteria

1. Configure a soft-delete policy and global configuration as shown in “Configuring Soft Deletes as a Global Configuration”.
2. Configure request criteria for soft deletes. The soft delete request criteria is configured for an external delete request from a member of the Internal Applications group matching an entry with object class "inetorgperson" with the request excluding the Soft Delete Request Control and the Hard Delete Request Control.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name "Soft Deletes" \
  --type simple \
  --set "description:Requests for soft delete" \
  --set operation-type:delete \
  --set operation-origin:external-request \
  --set "connection-criteria:Internal Applications" \
  --set not-all-included-request-control:1.3.6.1.4.1.30221.2.5.20 \
  --set "all-included-target-entry-filter:(objectClass=inetorgperson)"
```

3. In the soft delete policy created in step 1, set the `auto-soft-delete-connection-criteria` property to the simple criteria created in the previous step.

```
$ bin/dsconfig create-soft-delete-policy \
  --policy-name default-soft-delete-policy \
  --set "auto-soft-delete-request-criteria:Soft Deletes"
```

To Disable Soft Deletes by Request Criteria

- To disable soft deletes by request criteria, reset the soft-delete policy.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset auto-soft-delete-request-criteria
```

Configuring Soft Delete Automatic Purging

By default, the Directory Server retains soft-deleted entries indefinitely. For companies that want to set up automatic purging of soft-deleted entries, the server provides two properties on the Soft Delete Policy that can be configured for either the maximum retention time for all soft-deleted entries and/or the retained number of soft-deleted entries. These changes take effect without requiring a server restart.

To Configure Soft-Delete Automatic Purging

You can change either the retention time or the retained number of entries to enable automatic purging. By default, both are set to an indefinite retention time and number of entries. The time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), may be preceded by an integer to specify a quantity for that unit, such as "1 d", "52 w", etc. Once you configure the properties, the changes take effect immediately without the need for a server restart.

Note that the server will delete all of the soft-deleted entries according to the policy in effect. If the policy is changed while entries are in the process of being deleted, the new policy takes effect after the in-process batch of entries is deleted and applies to any remaining soft-deleted entries going forward according to the new policy.

1. Retrieve the name of the Soft Delete Policy in effect using the `dsconfig` command. For this example, the Soft Delete Policy is called `default-soft-delete-policy`.

```
$ bin/dsconfig get-global-configuration-prop \
  --property soft-delete-policy
```

2. Do one or both of the following:
   - Run `dsconfig` to set the retention time for soft-deleted entries.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set "soft-delete-retention-time:52 w"
```

- Run `dsconfig` to set the retained number of soft-deleted entries.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set soft-delete-retain-number-of-entries:1000000
```

3. The Soft Delete Policy must be assigned to the global configuration if it has not been assigned yet.

```
$ bin/dsconfig set-global-configuration-prop \
  --set soft-delete-policy:default-soft-delete-policy
```

To Disable Soft-Delete Automatic Purging

You can disable Soft-Delete automatic purging using the `dsconfig` command. The change takes effect immediately without the need of a server restart. However, if the server is in the middle of an automatic soft-delete purging, it may continue to purge entries until the next time it evaluates the Soft Delete Policy.

- Run `dsconfig` to reset the Soft-Delete Policy properties that control automatic purging: `soft-delete-retention-time` and `soft-delete-retain-number-of-entries`.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset soft-delete-retention-time \
  --reset soft-delete-retain-number-of-entries
```

Summary of Soft and Hard Delete Processed

The following table summarizes the resulting actions of a DELETE operation for soft deletes.

Table 7: If No Automatic Soft Delete Criteria is Configured

Action	Result
Delete	Performs a hard delete on the entry.
Delete with the Soft Delete Request Control	Performs a soft delete on the entry.
Delete of a soft-deleted entry	Performs a hard delete on the entry.
Delete of a soft-deleted entry with the Soft Delete Request Control	Not allowed. Generates an `UNWILLING_TO_PERFORM` error.
Delete with a Hard Delete Request Control	Performs a hard delete on the entry.
Delete of a soft-deleted entry with the Hard Delete Request Control.	Performs a hard delete on the entry.

The following table summarizes the resulting actions of a DELETE operation for soft deletes configured by connection criteria or request criteria.

Table 8: If Soft Delete Connection or Request Criteria is Configured

Action	Result
Delete not matching criteria	Performs a hard delete on the entry.
Delete matching criteria	Performs a soft delete on the entry.
Delete of a soft-deleted entry not matching criteria	Performs a hard delete on the entry.
Delete of a soft-deleted entry matching criteria	Performs a hard delete on the entry.
Delete not matching criteria with the Hard Delete Request Control	Performs a hard delete on the entry.
Delete matching criteria with the Hard Delete Request Control	Performs a hard delete on the entry.
Delete with Soft and Hard Delete Request Controls	Not allowed. Generates an `UNWILLING_TO_PERFORM` error.

Summary of Soft Delete Controls and Tool Options

The following table shows the OIDs for each soft delete control. The Soft Delete OIDs are defined in the LDAP SDK generated API documentation.

Table 9: SOft Delete OIDs

OID Type	OID
Soft Delete Request Control	1.3.6.1.4.1.30221.2.5.20
Soft Delete Response Control	1.3.6.1.4.1.30221.2.5.21
Hard Delete Request Control	1.3.6.1.4.1.30221.2.5.22
Soft Undelete Request Control	1.3.6.1.4.1.30221.2.5.23
Soft Delete Entry Access Control	1.3.6.1.4.1.30221.2.5.24

The following table shows the new tool options available for the Soft Delete operations.

Table 10: Soft Delete Tool Options

Action	Result
ldapdelete / ldapmodify	**--useSoftDelete/-s**. Process DELETE operations with the Soft Delete Request Control, whereby entries are renamed, and hidden instead of being permanently deleted. The Directory Server must be configured to allow soft deletes. Note that any entries in the LDIF file with the changetype of `delete` will be processed as a soft-delete request.

Action	Result
ldapdelete	**--useHardDelete**. Process DELETE operations with the Hard Delete Request Control, which bypasses any soft delete policies and processes the delete request immediately without retaining the entry as a soft-deleted entry. The Directory Server must be configured to allow soft deletes.
ldapsearch	**--includeSoftDeletedEntries {with-non-deleted-entries | without-non-deleted-entries | deleted-entries-in-undeleted-form}**. Process search operations with the Soft Delete Entry Access Control. Soft delete search options are as follows: • **with-non-deleted-entries**. Returns all entries matching the search criteria with the results including non-deleted and soft-deleted entries. • **without-non-deleted-entries**. Returns only soft-deleted entries matching the search criteria. • **deleted-entries-in-undeleted-form**. Returns only soft-deleted entries matching the search criteria with the results returned in their undeleted entry form. Users must have access to the Soft Delete Entry Access Control to be able to search for soft-deleted entries.
ldapmodify	**--allowUndelete**. Process ADD operations which include the `ds-undelete-from-dn` attribute as undelete requests. Undelete requests re-add previously soft-deleted entries back to the server as non-deleted entries by providing the Undelete Request Control with the ADD operation. The Directory Server must be configured to allow soft deletes to process any undelete requests and the client user must have the `soft-delete-read` privilege.

The following table shows the symbolic names that can be used with the server's LDAP commands using the --control/-J option.

Table 11: Soft Delete OID Symbolic Names using with the --control/-J Option

Control	Symbolic Name
Soft Delete Request Control	softdelete
Hard Delete Request Control	harddelete
Soft Undelete Request Control	undelete
Soft Delete Entry Access Control	softdeleteentryaccess

Monitoring Soft Deletes

The Directory Server provides monitoring entries and logs to track all soft delete operations. The access and debug logs do not have any options specific for soft deletes.

New Monitor Entries

Two new monitor entries are present for a backend monitor entry. Administrators will see the following additional monitor entries on `cn=userRoot Backend,cn=monitor`:

- **ds-soft-delete-entry-operations-count**. Displays the number of soft-deletes performed on the backend since server startup.
- **ds-undelete-operations-count**. Displays the number of undeletes performed on the backend since server startup.
- **ds-backend-soft-deleted-entry-count**. Displays the current number of soft-deleted entries in the database.
- **ds-auto-purged-soft-deleted-entry-count**. Displays the current number of soft-deleted entries purged since the backend or server was restarted.

To Monitor Soft Deletes

- Run `ldapsearch` on the `cn=userRoot Backend,cn=monitor` branch. Use a search criteria targeting the `ds-backend-monitor-entry` object class.

```
$ bin/ldapsearch --baseDN "cn=userRoot Backend,cn=monitor" \
  --searchScope sub "(objectclass=ds-backend-monitor-entry)"
```

```
dn: cn=userRoot Backend,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-backend-monitor-entry
objectClass: extensibleObject
cn: userRoot Backend
ds-backend-id: userRoot
ds-backend-base-dn: dc=example,dc=com
ds-backend-is-private: FALSE
ds-backend-entry-count: 200001
ds-backend-soft-deleted-entry-count: 1000
ds-soft-delete-operations-count: 40
ds-undelete-operations-count: 20
ds-auto-purged-soft-deleted-entry-count: 0
ds-base-dn-entry-count: 200001 dc=example,dc=com
ds-backend-writability-mode: enabled
```

Access Logs

The access log records the LDAP operations corresponding to soft delete and undelete for DELETE, SEARCH, MODIFY, and ADD operations with the related soft-deleted values. The access log does not require any configuration for soft delete.

For DELETE (soft-delete) operations, the access log displays:

```
[14/May/2012:09:40:16.942 -0500] DELETE RESULT conn=18 op=1 msgID=2
dn="uid=user.1,ou=People,dc=example,dc=com" resultCode=0 etime=30.367
softDeleteEntryDN="entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4+uid=user.1,
ou=People,dc=example,dc=com"
```

For SEARCH operations for soft-deleted entries, the log displays:

```
[14/May/2012:09:40:52.320 -0500] SEARCH RESULT conn=19 op=1 msgID=2
base="dc=example,dc=com" scope=2 filter="(objectclass=ds-soft-delete-entry)"
attrs="ALL" resultCode=0 etime=1.631 entriesReturned=1
```

For MODIFY operations of soft-deleted entries, the log displays:

```
[14/May/2012:09:42:43.679 -0500] MODIFY RESULT conn=20 op=1 msgID=1
dn="entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=exam-
ple,dc=com" resultCode=0 etime=2.639 changeToSoftDeletedEntry=true
```

For ADD (soft undelete) operations, the log displays:

```
[14/May/2012:09:58:16.728 -0500] ADD RESULT conn=25 op=1 msgID=1
dn="uid=user.0,ou=People,dc=example,dc=com" resultCode=0 etime=22.700
```

```
undeleteFromDN="entryUUID=ad55a34a-763f-358f-93f9-da86f9ecd9e4+uid=user.0,
ou=People,dc=example,dc=com"
```

Audit Logs

The audit log captures any MODIFY and DELETE operations of soft-deleted entries. These changes are recorded as fully commented-out audit log entries. The audit log does not require any configuration for soft deletes.

For any soft-deleted entry, the audit log entry displays the `ds-soft-delete-entry-dn` property and its soft-deleted entry DN.

```
# 14/May/2012:10:57:09.054 -0500; conn=30; op=1
# ds-soft-delete-entry-dn: entryUUID=68147342-1f61-3465-8489-
3de58c532130+uid=user.2,ou=People,dc=example,dc=com
dn: uid=user.2,ou=People,dc=example,dc=com
changetype: delete
```

For any MODIFY changes made, the log displays the LDIF, the modifier's name and update time.

```
# 14/May/2012:10:58:33.566 -0500; conn=33; op=1
# dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=People,dc=exam-
ple,dc=com
# changetype: modify
# replace: homePhone
# homePhone: +1 003 428 0966
#-
# replace: modifiersName
# modifiersName: uid=admin,dc=example,dc=com
#-
# replace: modifyTimestamp
# modifyTimestamp: 20131010020345.546Z
```

For any undelete of a soft-deleted entry, the log displays the `ds-undelete-from-dn` attribute plus the entry unique ID, create time and creator's name.

```
# 14/May/2012:10:59:21.754 -0500; conn=34; op=1
dn: uid=user.2,ou=People,dc=example,dc=com
changetype: add
uid: user.2
ds-undelete-from-dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=Peo-
ple,dc=example,dc=com
ds-entry-unique-id:: vw1jg801S7GWrTiS3UE5DA==
createTimestamp:: 20131010181148.630Z
creatorsName: uid=admin,dc=example,dc=com
```

For hard (permanent) deletes of a soft-deleted entry, the log displays the soft-deleted entry DN that was removed.

```
# 14/May/2012:11:00:14.055 -0500; conn=36; op=1
# dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=People,dc=exam-
ple,dc=com
# changetype: delete
```

To Configure the File-Based Audit Log for Soft Deletes

1. Enable the audit log if it is disabled.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
--set enabled:true
```

2. View the audit log. The `soft-delete-entry-audit-behavior` property is set to "`commented`" by default and provides additional information in comments about the soft-deleted entry that was either created or undeleted.

```
# 11/May/2012:15:33:17.552 -0500; conn=13; op=1
# ds-soft-delete-entry-dn:entryUUID=54716bfd-fbc4-3108-ac37-
bf6b1b166e37+uid=user.15,ou=People,dc=example,dc=com
dn: uid=user.15,ou=People,dc=example,dc=com
changetype: delete
```

Change Log

The change log can be configured to capture soft-delete changes to entries, so that external clients, such as a Data Sync Server, can access these changes. The `ds-soft-delete-entry` attribute represents an entry that has been soft-deleted and is part of the source entry passed into the changelog to indicate the entry has been soft-deleted.

Two important points about soft deletes and the changelog are as follows:

- All soft-delete operations appear in the changelog and appear as a regular DELETE operation. When a soft delete occurs, the resulting changelog entry will include a `ds-soft-delete-entry-dn` operational attribute with the value of the soft-deleted entry DN. If you are using the PingData Data Sync Server, it does recognize the `ds-soft-delete-entry-dn` attribute and does not do anything with it.
- The changelog backend's `soft-delete-entry-included-operation` property determines whether or not MODIFY or DELETE operations of soft-deleted entries appear in the changelog. By default, the property is not enabled by default.

To Configure Soft Deletes on the Changelog Backend

1. Configure soft deletes on the changelog backend.

```
$ bin/dsconfig set-backend-prop \
--backend-name changelog \
--set soft-delete-entry-included-operation:delete \
--set soft-delete-entry-included-operation:modify
```

2. Run a soft-delete operation on an entry.
3. View the changelog for the soft-deleted entry.

```
$ bin/ldapsearch --baseDN cn=changelog \
  "(objectclass=*)" "+"
```

```
dn: cn=changelog
subschemaSubentry: cn=schema
entryUUID: 9920f7e9-5a04-392a-82a8-32662d7d3863
ds-entry-checksum: 304022441
dn: changeNumber=1,cn=changelog
targetUniqueId: 94f634df-c90e-39aa-bd4a-9183c29746d0
changeTime: 20120511154141Z
ds-soft-delete-entry-dn: entryUUID=94f634df-c90e-39aa-bd4a-
9183c29746d0+uid=user.9,ou=People,dc=example,dc=com
modifyTimestamp: 20131010020345.546Z
createTimestamp:: 20131010181148.630Z
localCSN: 000001373C900852000000000003
modifiersName: uid=admin,dc=example,dc=com
entry-size-bytes: 298
subschemaSubentry: cn=schema
entryUUID: 459b06c6-89f3-307e-a515-22433eb420b6
createTimestamp: 20120511154141.431Z
modifyTimestamp: 20120511154141.431Z
ds-entry-checksum: 1157320579
```

Chapter

8

Importing and Exporting Data

Topics:

The PingDirectory Server supports import or export of the database backends in LDAP Data Interchange Format (LDIF). The `bin/import-ldif` and `bin/export-ldif` tools can be used to create or export database backends for online of offline servers. The tools support options that can restrict the input or output to a subset of the entries or a subset of the attributes within entries. The tools also provide features to compress, encrypt or digitally sign the data.

This chapter presents the following topics:

Importing Data

The PingDirectory Server provides initialization mechanisms to import database files. The `import-ldif` command-line tool imports data from an LDAP Data Interchange Format (LDIF) file. The data imported by the `import-ldif` command can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the LDIF file. The command also supports importing data that has been compressed, encrypted or digitally signed or both.

The `import-ldif` utility can be run with the server offline or online. If the server is online, administrators can initiate the import from a local or remote client. The LDIF file that contains the import data must exist on the server system. During an online import, the target database repository, or backend, will be removed from service and data held in that backend will not be available to clients.

The `import-ldif` tool has been modified to help guard against accidental overwriting of existing backend data with the addition of the --overwriteExistingEntries option. This option must be present when performing an import into a backend with a branch that already contains entries (although the option is not needed if a branch contains just a single base entry).

Validating an LDIF File

Prior to importing data, you can validate an import file using the Directory Server's `validate-ldif` tool. When run, the tool binds to the Directory Server, locally or remotely, and validates the LDIF file to determine whether it violates the server's schema. Those elements that do not conform to the schema will be rejected and written to standard output. You can specify the path to the output file to which the rejected entries are written and the reasons for their rejection. The `validate-ldif` tool works with regular non-compressed LDIF files or gzip-compressed LDIF files.

To process large files faster, you can also set the number of threads for validation. The tool also provides options to skip specified schema elements if you are only validating certain items, such as attributes only. Use the --help option to view the arguments.

To Validate an LDIF File

Content-Reference to:

- Use the `validate-ldif` tool to validate an LDIF file. Make sure the server is online before running this command.

```
$ bin/validate-ldif --ldifFile /path/to/data.ldif \
  --rejectFile rejectedEntries
```

```
1 of 200 entries (0 percent) were found to be invalid.
1 undefined attributes were encountered.
Undefined attribute departmentname was encountered 1 times.
```

Computing Database Cache Estimate

After successful completion of an import, the `import-ldif` command lists detailed information about the database cache characteristics of the imported data set. The current server configuration is considered along with the capabilities of the underlying hardware to guide decisions for changing JVM size and database-cache-percent for the backend.

The `import-ldif` command will complete with a summary of database cache usage characteristics for the imported data set. Additional files are available in the `/logs/tools` directory that describe the database cache characteristics in more detail.

Tracking Skipped and Rejected Entries

During import, entries can be skipped if they do not belong in the specified backend, or if they are part of an excluded base DN or filter. The --skipFile {path} argument can be used on the command line to indicate that any entries that are skipped should be written to a specified file. You can add a comment indicating why the entries were skipped.

Similarly, the --rejectFile {path} argument can be added to obtain information about which entries were rejected and why. An entry can be rejected if it violates the server's schema constraints, if its parent entry does not exist, if another entry already exists with the same DN, or if it was rejected by a plug-in.

Running an Offline Import

You can run the `import-ldif` tool offline to import LDIF data encoded with the UTF-8 character set. This data can come from LDIF files, compressed LDIF files (GZIP format), or from data generated using a MakeLDIF template. You do not need to authenticate as an administrator when performing offline LDIF imports.

To Perform an Offline Import

- Use the `import-ldif` command to import data from an LDIF file. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --rejectFile /path/to/reject.ldif --skipFile /path/to/skip.ldif
```

To Perform an Offline LDIF Import Using a Compressed File

- Use the `import-ldif` command to import data from a compressed gzip formatted file. You must also use the --isCompressed option to indicate that the input file is compressed. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --isCompressed \
  --ldifFile /path/to/data.gz --rejectFile /path/to/reject.ldif \
  --skipFile /path/to/skip.ldif
```

To Perform an Offline LDIF Import Using a MakeLDIF Template

- Use the `import-ldif` command to import data from a MakeLDIF template, which is located in the `<server-root>/config/MakeLDIF`. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

  The following command uses the standard data template and generates 10,000 sample entries, and then imports the file to the server.

```
$ bin/import-ldif --backendID userRoot \
  --templateFile config/MakeLDIF/example.template
```

Running an Online LDIF Import

Administrators can run LDIF imports while the server is online from another remote server. The online import resembles the offline import, except that you must provide information about how to connect and authenticate to the target server. You can schedule the import of an LDIF file to occur at a particular time using the --task and --start YYYYMMDDhhmmss options of the `import-ldif` tool.

You can also specify email addresses for users that should be notified whenever the import process completes (regardless of success or failure, or only if the import fails). To set up SMTP notifications, see Working with the SMTP Account Status Notification Handler.

To Perform an Online LDIF Import

- Use the `import-ldif` tool to import data from an LDIF. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task --hostname server1 --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --backendID userRoot --ldifFile userRoot.ldif
```

To Schedule an Online Import

1. Use the `import-ldif` tool to import data from an LDIF file at a scheduled time. To specify a time in the UTC time zone, include a trailing “Z”. Otherwise, the time will be treated as a local time in the time zone configured on the server. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task \
  --hostname server1 \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password \
  --backendID userRoot \
  --ldifFile /path/to/data.ldif \
  --start 20111026010000 \
  --completionNotify import-complete@example.com \
  --errorNotify import-failed@example.com
```

```
Import task 2011102617321510 scheduled to start Oct 26, 2011 1:00:00 AM CDT
```

2. Confirm that you successfully scheduled your import task using the `manage-tasks` tool to view a summary of all tasks on the system.

```
$ bin/manage-tasks --summary
```

```
ID              Type   Status
-----------------------------------------------
2011102617321510 Import Waiting on start time
```

3. Use the `manage-tasks` tool to monitor the progress of this task. Use the task ID of the import task. If you cannot find the task ID, use the --summary option to view a list of all tasks scheduled on the Directory Server.

```
$ bin/manage-tasks --info 2011102617321510
```

```
Task                    Details
-----------------------------------------------------
ID                      2011102617321510
Type                    Import
Status                  Waiting on start time
Scheduled Start Time    Oct 26, 2011 1:00:00 AM CDT
Actual Start Time
Completion Time
Dependencies Failed     None
Dependency Action       None
Email Upon Completion   admin@example.com
Email Upon Error        admin@example.com

Import        Options
----------------------------
LDIF File  /path/to/data.ldif
Backend ID userRoot
```

To Cancel a Scheduled Import

- Use the `manage-tasks` tool to cancel the scheduled task.

```
$ bin/manage-tasks --cancel 2011102417321510
```

Adding Entries to an Existing Directory Server

To add entries to an existing Directory Server while preserving operational attributes, such as `createTimestamp` or `modifiersName`, the Ignore No User Modification control must be attached to the request. The Ignore No User Modification control allows modification of certain attributes that have the No User Modification constraint. Special care should be used with this control.

The Ignore No User Modification is only applied to ADD requests. Using the control to modify an existing entry, resulting in an operational attribute change, will fail.

To Append Entries to an Existing Directory Server

- Use `ldapmodify` with the Ignore No User Modification control (i.e., the OID is 1.3.6.1.4.1.30221.2.5.5).

```
$ bin/ldapmodify --control 1.3.6.1.4.1.30221.2.5.5 \
  --filename change-record.ldif
```

Filtering Data Import

The `import-ldif` command provides a way to either include or exclude specific attributes or entries during an import. The arguments are summarized as follows:

Table 12: Inclusion and Exclusion Arguments for import-ldif

Argument	Description
--includeBranch	Base DN of a branch to include in the LDIF import (can be specified multiple times)
--excludeBranch	Base DN of a branch to exclude from the LDIF import (can be specified multiple times)
--includeAttribute	Attribute to include in the LDIF import (can be specified multiple times)
--excludeAttribute	Attribute to exclude from the LDIF import (can be specified multiple times)
--includeFilter	Search filter to identify entries to include in the LDIF import (can be specified multiple times)
--excludeFilter	Search filter to identify entries to exclude from the LDIF import (can be specified multiple times)
--excludeOperational	Exclude operational attributes from the LDIF import.
--excludeReplication	Exclude replication attributes from the LDIF import.
--excludeSoftDelete	Exclude soft delete entries from the LDIF import.

Exporting Data

The PingDirectory Server `export-ldif` command-line tool exports data from Directory Server backend to an LDAP Data Interchange Format (LDIF) file. The tool must be run in the non-task based mode, which implies that it works outside of the server JVM process. The `export-ldif` must be run without connection or task arguments while the server is either online or offline. This tool exports a point-in-time snapshot of the backend which is guaranteed to provide a consistent state of the database, in LDIF, which can be reimported with `import-ldif` if necessary.

The data exported by `export-ldif` can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the backend. This is accomplished by specifying branches, filters, and attributes to include or exclude. The exported LDIF can be compressed, encrypted or digitally signed.

👉 **Note:** LDIF exports can be configured as recurring tasks with `dsconfig create-recurring-task`, and then scheduled to run when added to a recurring task chain.

To Perform an Export

- Use the `export-ldif` command to export data to an LDIF file.

```
$ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif
```

To Perform an Export from Specific Branches

- Use the `export-ldif` command to export data to an LDIF file under a specific branch from the userRoot backend of the local Directory Server into a compressed file. The command also excludes operational attributes from the exported data and wraps long lines at column 80.

```
$ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif.gz --
compress \
  --includeBranch ou=people,dc=example,dc=com --excludeOperational \
  --wrapColumn 80
```

Encrypting LDIF Exports and Signing LDIF Files

The Directory Server provides features to encrypt data during an LDIF export using the `export-ldif` --encryptLDIF option and to allow the encrypted LDIF file to be imported onto the same instance or another server in the same replication topology using the `import-ldif` tool. A `--doNotEncrypt` argument can be used to force an LDIF export to be unencrypted, even if automatic encryption is enabled. The `--maxMegabytesPerSecond` argument can be used to impose a limit on the rate at which the LDIF file may be written to disk.

The `export-ldif` tool can be used with the `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, and `--encryptionSettingsDefinitionID` arguments to specify which key to use for encrypting the export. The `import-ldif tool` will automatically detect encryption and compression, and have `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile` options as well.

The Directory Server also provides an additional argument that digitally signs the contents of the LDIF file, which ensures that the content has not been altered since the export. To digitally sign the contents of the exported LDIF file, use the `export-ldif` --sign option. To allow a signed LDIF file to be imported onto the same instance or another server in the same topology, use the `import-ldif` --isSigned option.

Note that there is not much added benefit to both signing and encrypting the same data, since encrypted data cannot be altered without destroying the ability to decrypt it.

To Encrypt an LDIF Export

- Run `export-ldif` tool with the `--encryptLDIF` option to encrypt the data during an export to an output LDIF file. The following command runs an offline export of the userRoot backend, and encrypts the file when written to an output file called `data.ldif`.

```
$ bin/export-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --encryptLDIF
```

To Import an Encrypted LDIF File

An encrypted LDIF file can be imported into the same instance from which it was exported, or into any other server in the same replication topology with that instance. You cannot import an encrypted LDIF file into a server that is not in some way connected to the instance from which it was exported.

- Run the import-ldif tool to import the encrypted LDIF file from the previous example. The command imports the `data.ldif` file, decrypts the contents while overwriting the existing contents to the `userRoot` backend. The tool automatically determines encryption and compression, and it can automatically identify the correct key for exports that were encrypted with a key obtained from an encryption settings definition or an internal topology key.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries
```

To Sign an Export

- Run `export-ldif` tool with the `--sign` option to digitally sign the data during an export to an output LDIF file. The following command runs an offline export of the `userRoot` backend, and signs the content when written to an output file called `data.ldif`.

```
$ bin/export-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif --sign
```

To Import a Signed LDIF File

- Run the `import-ldif` tool to import the signed LDIF file from the previous example. The command imports the `data.ldif` file, checks the signature of the contents while overwriting the existing contents to the `userRoot` backend. The command requires the --isSigned option, which instructs the tool that the contents of the LDIF file is signed.

```
$ bin/import-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries --isSigned
```

Filtering Data Exports

The `export-ldif` command analogous arguments to the `import-ldif` tool to provide a way to either include or exclude specific attributes or entries during an export. The arguments are summarized as follows:

Table 13: Inclusion and Exclusion Arguments for export-ldif

Argument	Description
--includeBranch	Base DN of a branch to include in the LDIF export (can be specified multiple times)
--excludeBranch	Base DN of a branch to exclude from the LDIF export (can be specified multiple times)
--includeAttribute	Attribute to include in the LDIF export (can be specified multiple times)
--excludeAttribute	Attribute to exclude from the LDIF export (can be specified multiple times)
--includeFilter	Filter to identify entries to include in the LDIF export (can be specified multiple times)
--excludeFilter	Filter to identify entries to exclude from the LDIF export (can be specified multiple times)
--excludeOperational	Exclude operational attributes from the LDIF export.
--excludeReplication	Exclude replication attributes from the LDIF export.
--excludeSoftDelete	Exclude soft delete entries from the LDIF export.

Scrambling Data Files

The Directory Server `transform-ldif` tool provides backward compatibility with the former `scramble-ldif` tool, with additional functionality for configuring input and output files. The `transform-ldif` tool reads data from one or more source LDIF files and writes the transformed data to a single output file.

Using this tool to scramble data, enables obscuring the values of certain attributes so that it is difficult to determine the original values in the source data, while also preserving the characteristics of the associated attribute syntax. This process is repeatable, so that if the same value appears multiple times, it will yield the same scrambled representation each time. Scrambling can be applied to both LDIF entries and LDIF change records.

The process of scrambling data is not the same as encryption. It should only be used to provide simple obfuscation of data. The following are general guidelines for scrambling attributes:

- If the attribute is `userPassword` and its value starts with a scheme name surrounded by curly braces, such as "{SSHA256}XrgyNdl3fid7KYdhd/Ju47KJQ5PYZqlUlyzxQ28f/QXUnNd9fupj9g==", the scheme name will be left unchanged and the rest of the value will be treated like a generic string.
- If the attribute is `authPassword` and its value contains at least two dollar signs, such as "SHA256$QGbHtDCi1i4=$8/X7XRGaFCovC5mn7ATPDYlkVoocDD06Zy3lbD4AoO4=", the portion up to the first dollar sign (which represents the name of the encoding scheme) is preserved and the remainder of the value is treated like a generic string.
- If an attribute has a Boolean syntax, the scrambled value will be either `TRUE` or `FALSE`. The determination to use a value of `TRUE` or `FALSE` is random, so scrambling Boolean values is not repeatable. By randomizing the scrambling for Boolean values, the syntax and obfuscation of the original value is preserved.
- If an attribute has a distinguished name syntax (or a related syntax, such as a name and optional UID), scrambling is applied to the values of RDN components for any attributes to be scrambled. For example, if the tool is configured to scramble both the `member` and `uid` attributes, and an entry has a member attribute with a value of "uid=john.doe,ou=People,dc=example,dc=com", that member value will be scrambled in a way that only obscures the "john.doe" portion but leaves the attribute names and all values of non-scrambled attributes intact.
- If an attribute has a generalized time syntax, that value is replaced with a randomized timestamp using the same format (the same number of digits and the same time zone indicator). The randomization will be over a time range that is double the difference between the time the `transform-ldif` tool was launched and the timestamp to be scrambled. For values where that time difference is less than one day, one day will be added to the difference before it is doubled.
- If an attribute has an integer, numeric string, or telephone number syntax, scrambling is only applied to numeric digits while all other characters are left intact. If there are multiple digits, then the first digit will be nonzero.
- If an attribute has an octet string syntax, it is scrambled as follows:
  - Each byte that represents a lowercase ASCII letter is replaced with a randomly-selected lowercase ASCII letter.
  - Each byte that represents a uppercase ASCII letter is replaced with a randomly-selected uppercase ASCII letter.
  - Each byte that represents an ASCII digit is replaced with a randomly-selected ASCII digit.
  - Each byte that represents a printable ASCII symbol is replaced with a randomly-selected printable ASCII symbol.
  - Each byte that represents an ASCII control character is replaced with a randomly-selected ASCII letter, digit, or symbol.
  - Each non-ASCII byte will be replaced with a randomly-selected non-ASCII byte.
- If an attribute has a value that represents a valid JSON object, the resulting value will also be a JSON object. All field names will be left intact, and only the values of those fields may be scrambled. If the `--scrambleJSONField` argument is provided, only the specified fields will have values scrambled. Otherwise, the values of all fields will be scrambled. Field values are scrambled as follows:
  - Null values are not scrambled.
  - Boolean values are replaced with randomly-selected Boolean values. As with attributes with a Boolean syntax, these values are non-repeatable.

  - Number values will have only their digits replaced with randomly-selected digits and all other characters (minus sign, decimal point, exponentiation indicator) are left unchanged.
  - String values will be replaced with a randomly-selected generic string.
  - Array values have scrambling applied as appropriate for each value in the array. If the array field itself should be scrambled, then all values in the array are scrambled. Otherwise, only JSON objects contained inside the array have scrambling applied to appropriate fields.
  - JSON values have scrambling applied as appropriate for their fields.
- If an attribute does not match any of the previous criteria, it is scrambled as follows:

  - Each lowercase ASCII letter is replaced with a randomly-selected lowercase ASCII letter.
  - Each uppercase ASCII letter replaced with a randomly-selected uppercase ASCII letter.
  - Each ASCII digit is replaced with a randomly-selected ASCII digit.
  - All other characters are left unchanged.

The following example reads from an LDIF file named `original.ldif`, scrambles the values of the `telephoneNumber`, `mobile`, and `homeTelephoneNumber` attributes, and writes the results to `scrambled.ldif`:

```
$ bin/transform-ldif --sourceLDIF original.ldif \
  --targetLDIF scrambled.ldif \
  --scrambleAttribute telephoneNumber \
  --scrambleAttribute mobile \
  --scrambleAttribute homeTelephoneNumber \
  --randomSeed 0
```

Chapter 9

Backing Up and Restoring Data

Topics:

The PingDirectory Server provides efficient `backup` and `restore` command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based cron scheduler or using the Directory Server's Task-based scheduler.

This chapter presents the following topics:

Backing Up and Restoring Data

Administrators should have a comprehensive backup strategy and schedule that comprise of daily, weekly, and monthly backups including incremental and full backups of the directory server data, configuration, and backends. Administrators should also have a backup plan for the underlying filesystem. This dual purpose approach provides excellent coverage in the event that a server database must be restored for any reason.

Note: Backups can be configured as recurring tasks with `dsconfig create-recurring-task`, and then scheduled to run when added to a recurring task chain.

The PingDirectory Server provides efficient `backup` and `restore` command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based cron scheduler or using the Directory Server's Task-based scheduler. The Directory Server can run backups with the server online while processing other requests, so that there is no need to shut down the server or place it in read-only mode prior to starting a backup.

If you back up more than one backend, the `backup` tool creates a subdirectory below a specified backup directory for each backend. If you back up only a single backend, then the backup files will be placed in the specified directory. A single directory can only contain files from one backend, so that you cannot have backup files from multiple different backends in the same backup directory.

When performing a backup, the server records information about the current state of the server and backend, including the server product name, the server version, the backend ID, the set of base DNs for the backend, and the Java class used to implement the backend logic. For JE backends, the backup descriptor also includes information about the Berkeley DB JE version and information about the attribute and VLV indexes that have been defined.

When restoring a backup, the server compares the descriptor obtained from the backup with the current state of the server and backend. If any problems are identified, the server generates warnings or errors. The administrator can choose to ignore the warnings with the `ignoreCompatibilityWarnings` option to the `restore` tool, whereas errors will always cause the restore to fail. For example, when restoring a *newer* backup into an older version of the server, a warning will be generated. When restoring an *older* backup into a new version of the server, no warning will be generated, but because the `config` and `schema` backends require special handling, the server generates an error if the server versions do not match exactly (major, minor, point, and patch version numbers).

Both the `backup` and `restore` tools provide encryption options `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, and `--encryptionSettingsDefinitionID` that can be used to specify which key to use for encrypting the backup. For backups encrypted with an encryption settings definition or an internal topology key, the server will automatically determine the correct key. Or, the `--doNotEncrypt` argument can be used to force a backup to be unencrypted even if automatic encryption is enabled.

If needed, the `--maxMegabytesPerSecond` argument can be used to impose a limit on the rate at which the backup may be written to disk.

Retaining Backups

The backup tool can be used with either the `--retainPreviousFullBackupCount` or `--retainPreviousFullBackupAge` arguments to identify which previous backups should be preserved. Any other backups in that directory will be removed. A new backup will always be preserved. If the new backup is an incremental backup, then any other backups it depends on will also be preserved. However, older backups in the same directory are eligible to be removed.

If the `--retainPreviousFullBackupCount` argument is provided, that number of the most recent previous full backups will be preserved (along with any incremental backups that depend on them). Any other previous full backups (and their dependent incremental backups) can be removed. If the `--retainPreviousFullBackupAge` argument is provided, its value must be a duration represented as an integer followed by a time unit. Any full backups (and their dependent incremental backups) created longer ago than that duration will be eligible to be removed. If both the `--retainPreviousFullBackupCount` and `--retainPreviousFullBackupAge` arguments are provided, then only backups that don't satisfy either condition will be deleted. A value of zero can be specified for the `--retainPreviousFullBackupCount` argument so that only the most recent backup is preserved (along with its dependencies), and all previous backups will be removed.

Note: The `remove-backup` tool also supports the `--retainFullBackupCount` and `--retainFullBackupAge` arguments to delete any backups outside the provided retention criteria.

To List the Available Backups on the System

- Use the `restore` tool to list the backups in a `backup` directory.

```
$ bin/restore --listBackups --backupDirectory /mybackups
```

```
[13:26:21] The console logging output is also available in '/
ds/PingDirectory/logs/ tools/restore.log'

Backup ID:          20120212191715Z
Backup Date:        12/Feb/2012:13:17:19 -0600
Is Incremental:     false
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:    false
Dependent Upon:     none
Backup ID:          20120212192411Z
Backup Date:        12/Feb/2012:13:24:16 -0600
Is Incremental:     true
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:    false
Dependent Upon:     20120212191715Z
```

To Back Up All Backends

- Use `backup` to save the all of the server's backends. The optional --compress option can reduce the amount of space that the backup consumes, but can also significantly increase the time required to perform the backup.

```
$ bin/backup --backUpAll --compress --backupDirectory /path/to/backup
```

To Back Up a Single Backend

- Go to the server root directory, and use the `backup` tool to save the single backend, userRoot.

```
$ bin/backup --backendID userRoot --compress --backupDirectory /path/to/
backup
```

To Perform an Offline Restore

- Use the `restore` command to restore the userRoot backend. Only a single backend can be restored at a time. The Directory Server must be shut down before performing an offline restore.

```
$ bin/restore --backupDirectory /path/to/backup/userRoot
```

Note: The server root directory should never be restored from a file system backup or snapshot.

To Assign an ID to a Backup

- Go to the server root directory, and use the backup tool to save the single backend, userRoot. The following command assigns the backup ID "weekly" to the userRoot backup. The backup file appears under `backups/userRoot` directory as `userRoot-backup-weekly`.

```
$ bin/backup --backupDirectory /path/to/backups/userRoot \
  --backendID userRoot --backupID weekly
```

To Run an Incremental Backup on All Backends

The Directory Server provides support for incremental backups, which backs up only those items that have changed since the last backup (whether full or incremental) on the system, or since a specified earlier backup. Incremental backups must be placed in the same backup directory as the full backup on which they are based.

Not all backends support incremental backups. If a backend does not support incremental backups, use of the `--incremental` option will have no effect, and a full backup will be taken.

- The following command runs an incremental backup on all backends based on the most recent backup:

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup
```

To Run an Incremental Backup on a Single Backend

- Go to the server root directory, and use `backup` to save the single backend, userRoot.

```
$ bin/backup --backendID userRoot --incremental --backupDirectory /path/to/
backup
```

To Run an Incremental Backup based on a Specific Prior Backup

- You can run an incremental backup based on a specific prior backup that is not the most current version on the system. To get the backup ID, use the `restore` --listBackups command (see below).

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup \
  --incrementalBaseID backup-ID
```

To Restore an Incremental Backup

The process for restoring an incremental backup is exactly the same as the process for restoring a full backup for both the online and offline restore types. The `restore tool` will automatically ensure that the full backup and any intermediate incremental backups are restored first before restoring the final incremental backup. The tool will not restore any files in older backups that are no longer present in the final data set.

To Schedule an Online Backup

You can schedule a backup to run as a Task by specifying the timestamp with the --task and --start options. The option is expressed in "YYYYMMDDhhmmss'" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

For online (remote) backups, the backup operation can be conducted while the PingDirectory Server is online if you provide information about how to connect and to authenticate to the target Directory Server.

- You can schedule the backup to occur at a specific time using the Task-based --start YYYYMMDDhhmmss option. To specify a time in the UTC time zone format, add a trailing “Z” to the time. Otherwise, the time will be treated as a local time in the time zone configured on the server.

```
$ bin/backup --backUpAll --task --start 20111025010000 \
  --backupDirectory /path/to/backup --completionNotify admin@example.com \
  --errorNotify admin@example.com
```

```
Backup task 2011102500084110 scheduled to start Oct 28, 2011 1:00:00 AM CDT
```

To Schedule an Online Restore

By providing connection and authentication information (and an optional start time), the restore can be performed via the Tasks subsystem while the server is online. The Tasks subsystem allows you to schedule certain operations, such as `import-ldif`, `backup`, `restore`, `start-server`, and `stop-server`. You can schedule a restore to run as a Task by specifying the timestamp with the --task and --start options. The option is expressed in "YYYYMMDDhhmmss'" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

- The backend that is being restored will be unavailable while the restore is in progress. To specify a time in the UTC time zone, add a trailing "Z" to the time. Otherwise, the time will be treated as a local time in the configured time zone on the server.

```
$ bin/restore --task --start 20111025010000 \
  --backupDirectory /path/to/backup/userRoot \
  --completionNotify admin@example.com --errorNotify admin@example.com
```

To Encrypt a Backup

- Go to the server root directory, and use the `backup` tool to backup up the single backend, userRoot and encrypt it with the `--encrypt` option.

```
$ bin/backup --encrypt --backendID userRoot --compress --backupDirectory /
path/to/backup
```

To Sign a Hash of the Backup

- Go to the server root directory, and use the `backup` tool to backup up the single backend, userRoot. Use the `–signHash` option to generate a hash of the backup contents and digitally sign the hash of the backup contents. If you want to generate only a hash of the backup contents, run backup with the `--hash` option.

```
$ bin/backup --signHash --backupDirectory backups/userRoot --backendID
 userRoot \
  --backupDirectory /path/to/backup
```

To Restore a Backup

- Go to the server root directory, and use the `restore` tool to restore a backup. The `backup` tool uses a descriptor file to access property information used for the backup, indicating if the backup was compressed, signed and/or encrypted.

```
$ bin/restore --backupDirectory /path/to/backup
```

Moving or Restoring a User Database

Part of any disaster recovery involves the restoration of the user database from one server to another. You should have a well-defined backup plan that takes into account whether or not your data is replicated among a set of servers. The plan is the best insurance against significant downtime or data loss in the event of unrecoverable database issue.

Keep in mind the following general points about database recovery:

- **Regular Backup from Local Replicated Directory Server**. Take a backup from a local replicated directory server and restore to the failed server. This will be more recent than any other backup you have.
- **Restore the Most Recent Backup**. Restore the most recent backup from a local server. In some cases, this may be preferred over taking a new backup if that would adversely impact performance of the server being backed up although it will take longer for replication to play back changes.

- **Contact Support**. If all else fails, contact your authorized support provider and they can work with you (and possibly in cooperation with the Oracle Berkeley DB JE engineers) to try a low-level recovery, including tools that attempt to salvage whatever data they can obtain from the database.

Comparing the Data in Two Directory Servers

The PingDirectory Server provides an `ldap-diff` tool to compare the data on two LDAP servers to determine any differences that they may contain. The differences are identified by first issuing a subtree search on both servers under the base DN using the default search filter `(objectclass=*)` to retrieve the DNs of all entries in each server. When the tool finds an entry that is on both servers, it retrieves the entry from each server and compares all of its attributes. The tool writes any differences it finds to an LDIF file in a format that could be used to modify the content of the source server, so that it matches the content of the target server. Any non-synchronized entries can be compared again for a configurable number of times with an optional pause between each attempt to account for replication delays.

You can control the specific entries to be compared with the --searchFilter option. In addition, only a subset of attributes can be compared by listing those attributes as trailing arguments of the command. You can also exclude specific attributes by prepending a ^ character to the attribute. (On Windows operating systems, excluded attributes must be quoted, for example, "`^attrToExclude`".) The `@objectClassName` notation can be used to compare only attributes that are defined for a given objectclass.

The `ldap-diff` tool can be used on servers actively being modified by checking differing entries multiple times without reporting false positives due to replication delays. By default, it will re-check each entry twice, pausing two seconds between checks. These settings can be configured with the --numPasses and --secondsBetweenPass options. If the utility cannot make a clean comparison on an entry, it will list any exceptions in comments in the output file.

The Directory Server user specified for performing the searches must be privileged enough to see all of the entries being compared and to issue a long-running, unindexed search. For the Directory Server, the out-of-the-box `cn=Directory Manager` user has these privileges, but you can assign the necessary privileges by setting the following attributes in the user entry:

```
ds-cfg-default-root-privilege-name: unindexed-search
ds-cfg-default-root-privilege-name: bypass-acl
ds-rlim-size-limit: 0
ds-rlim-time-limit: 0
ds-rlim-idle-time-limit: 0
ds-rlim-lookthrough-limit: 0
```

The `ldap-diff` tool tries to make efficient use of memory, but it must store the DNs of all entries in memory. For Directory Servers that contain hundreds of millions of entries, the tool might require a few gigabytes of memory. If the progress of the tool slows dramatically, it might be running low on memory. The memory used by the `ldap-diff` tool can be customized by editing the `ldap-diff.java-args` setting in the `config/java.properties` file and running the `dsjavaproperties` command.

If you do not want to use a subtree search filter, you can use an input file of DNs for the source, target, or both. The format of the file can accept various syntaxes for each DN:

```
dn: cn=this is the first dn
dn: cn=this is the second dn and it is wrapped cn=this is the third dn
# The following DN is base-64 encoded dn::
Y249ZG9uJ3QgeW91IGhhdmUgYmV0dGVyIHRoaW5ncyB0byBkbyB0aGFuIHNlZSB3aGF0IHRoaXMgc2F5cw==
# There was a blank line above dn: cn=this is the final entry.
```

CAUTION: Do not manually update the servers when the tool identifies differences between two servers involved in replication. First contact your authorized support provider for explicit confirmation, because manual updates to the servers risk introducing additional replication conflicts.

To Compare Two Directory Servers Using ldap-diff

1. Use `ldap-diff` to compare the entries in two Directory Server instances. Ignore the `userpassword` attribute due to the one-way password hash used for the password storage scheme.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN dc=example,dc=com --searchFilter "(objectclass=*)"
 "^userpassword"
```

2. Open the output file in a text editor to view any differences. The file is set up so that you can re-apply the changes without any modification to the file contents. The file shows any deletes, modifies, and then adds from the perspective of the source server as the authoritative source.

```
# This file contains the differences between two LDAP servers.
#
# The format of this file is the LDIF changes needed to bring server
# ldap://server1.example.com:1389 in sync with server
# ldap://server2.example.com:2389.
#
# These differences were computed by first issuing an LDAP search at both
# servers under base DN dc=example,dc=com using search filter
 (objectclass=*)
# and search scope SUB to first retrieve the DNs of all entries. And then
 each
# entry was retrieved from each server and attributes: [^userpassword] were
# compared. # # Any entries that were out-of-sync were compared a total of 3
 times
# waiting a minimum of 2 seconds between each attempt to account for
 replication
# delays.
#
# Comparison started at [24/Feb/2010:10:34:20 -0600]
# The following entries were present only on ldap://server2.example.com:2389
 and
# need to be deleted. This entry existed only on ldap://
server1.example.com:1389
# Note: this entry might be incomplete. It only includes attributes:
# [^userpassword]dn: uid=user.200,ou=People,dc=example,dc=com
# objectClass: person
# objectClass: inetOrgPerson
... (more attributes not shown) ...
# st: DC
dn: uid=user.200,ou=people,dc=example,dc=com
changetype: delete

# The following entries were present on both servers but were out of sync.

dn: uid=user.199,ou=people,dc=example,dc=com
changetype: modify
add: mobile
mobile: +1 300 848 9999
-
delete: mobile
mobile: +1 009 471 1808

# The following entries were missing on ldap://server2.example.com:2389 and
 need
# to be added. This entry existed only on ldap://server2.example.com:2389
# Note: this entry might be incomplete. It only includes attributes:
```

```
# [^userpassword]
dn: uid=user.13,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
... (more attributes not shown) ...
# Comparison completed at [24/Feb/2010:10:34:25 -0600]
```

To Compare Configuration Entries Using ldap-diff

- Use `ldap-diff` to compare the configuration entries in two Directory Server instances. The filter searches all configuration entries. Ignore the `userpassword` attribute due to the password storage scheme that uses a one-way hashing algorithm.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN cn=config --searchFilter "(objectclass=*)" "^userpassword"
```

To Compare Entries Using Source and Target DN Files

- Use `ldap-diff` to compare the entries in two Directory Server instances. In the following example, the utility uses a single DN input file for the source and target servers, so that no search filter is used. Ignore the `userpassword` attribute due to the password storage scheme that uses a one-way hashing algorithm.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN "dc=example,dc=com" --sourceDNsFile input-file.ldif \
  --targetDNsFile input-file.ldif "^userpassword"
```

To Compare Directory Servers for Missing Entries Only Using ldap-diff

- Use `ldap-diff` to compare two Directory Servers and return only those entries that are missing on one of the servers using the --missingOnly option, which can significantly reduce the runtime for this utility.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN dc=example,dc=com --searchFilter "(objectclass=*)"
 "^userpassword" \
  --missingOnly
```

Chapter 10

Working with Indexes

Topics:

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server.

This chapter presents topics related to indexes:

Overview of Indexes

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server. The Directory Server automatically creates index files when you first initialize a base DN or when you use the `dsconfig` tool to create a local DB backend. During modify operations, the Directory Server updates the database index files. If encryption is enabled on the server, indexes will be encrypted.

The PingDirectory Server comes with the following types of indexes:

- **Default system indexes** to ensure that the server operates efficiently. Indexes consist of database files that contain index keys mapping to the list of entry IDs.
- **Default Local DB indexes** that are created for each database suffix. Modify the index to meet your system's requirements using the `dsconfig` tool.
- **Local DB VLV indexes** that allow a client to request the server to send search results using the Virtual List View control.
- **Filtered Indexes** that provide the ability to index an attribute but only for entries that match a specified filter based on an equality index. The filtered index can only be used for searches containing that filter. The filtered index can be maintained independently of the equality index for that attribute and even if a normal equality index is not maintained for that attribute.

General Tips on Indexes

Administrators should keep the following tips in mind when working with indexes:

- **Important Critical Indexes**. The Directory Server has several built-in indexes on the Local DB Backend that are critical to internal server processing and should never be removed.

  ```
  aci, ds-entry-unique-id, objectClass
  ```
- **Built-in Indexes for Efficient Queries**. The Directory Server has built-in indexes on the Local DB Backend. Internal processing of the server relies on the `aci`, `ds-soft-delete-from-dn`, `ds-soft-delete-timestamp`, `entryUUID`, `member`, `objectClass`, and `uniqueMember` indexes, which must not be removed. The `mail` and `uid` indexes can be removed, but these attributes are referenced from the Password Modify Extended Operation and will cause problems with components such as the Exact Match Identity Mapper. If the `mail` or `uid` indexes are removed, additional configuration changes may be necessary to ensure that the server starts properly. The `cn`, `givenName`, `mail`, `sn`, and `telephoneNumber` indexes can be safely removed if clients do not query on these attributes. This will reduce the size of the database both on disk and in memory.
- **Online Rebuilds**. Whenever an online index rebuild is in progress, the data in that backend will be available and writable although the index being rebuilt will not be used; therefore, searches which attempt to use that attribute might be unindexed.
- **Index Rebuild Administrative Alert**. The Directory Server generates an administrative alert when the rebuild process begins and ends. It will have a degraded-alert-type of "index-rebuild-in-progress" so that a Directory Proxy Server, such as the Directory Server can avoid using that server while the rebuild is in progress.
- **System Indexes Cannot be Rebuilt**. The contents of the backend must be exported and re-imported in order to rebuild system indexes. See the table below for the list of system indexes.
- **Indexing Certain Attributes**. You should ensure that the following recommendations are used when setting up the indexes.
  - Equality and substring indexes should not be used for attributes that contain binary data.
  - Approximate indexes should be avoided for attributes containing numbers, such as telephone numbers.
- **Unindexed Searches**. Unindexed attributes result in longer search times as the database itself has to be searched instead of the database index file. Only users with the `unindexed-search` privilege are allowed to carry out unindexed searches. In general, applications should be prevented from performing unindexed searches, so that

searches that are not indexed would be rejected rather than tying up a worker thread. The ways to achieve this include:

- Make sure that only the absolute minimum set of users have the `unindexed-search` privilege. This privilege can be used without any other restrictions.
- To allow unindexed searches with some control, the Permit Unindexed Search request control can be used with the `unindexed-search-with-control` privilege. With this privilege, a user will only be permitted to request an unindexed search if the search request includes the Permit Unindexed Search request control. The `unindexed-search privilege` allows a client to request an unindexed search without this control.
- The Reject Unindexed Search request control can be used to explicitly indicate that a client does not want the server to process an unindexed search request, regardless of priviledges. See the LDAP SDK for information about these controls. These capabilities are also available with the `ldapsearch` tool.
- Make sure that `allow-unindexed-searches` property is set to false in all client connection policies, in which unindexed searches should never be necessary. If the client connection policy should allow undindexed searches, set the `allow-unindexed-searches-with-control` property to true. If `allow-unindexed-searches` is false but, `allow-unindexed-searches-with-control` is true, the policy will only permit an unindexed search if the request includes the Permit Unindexed Search request control. See the LDAP SDK and the `ldapsearch` tool for more information.
- Set a nonzero value for the `maximum-concurrent-unindexed-searches` global configuration property to ensure that if unindexed searches are allowed, only a limited number of them will be active at any given time. Administrators can configure the maximum number of concurrent unindexed searches by setting a property under Global Configuration.

  To change the maximum number of concurrent unindexed searches, use the `dsconfig` tool to set a value for the number. A value of "0" (default) represents no limit on the number of concurrent unindexed searches.

  ```
  $ bin/dsconfig set-global-configuration-prop \
    --set maximum-concurrent-unindexed-searches:2
  ```

- **Index Entry Limit**. The Directory Server specifies an index entry limit property. This property defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (default value is 4000), then you must rebuild the indexes using the `rebuild-index` tool. If an index entry limit value is set for the local DB backend, it overrides the value set for the overall JE backend index entry limit configuration (i.e., 4000).

  To change the default index entry limit, use the `dsconfig` tool as seen in the following example:

  ```
  $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
    --index-name cn --set index-entry-limit:5000
  ```

- **Rebuild Index vs Full Import**. You can expect a limited amount of database growth due to the existence of old data when running `rebuild-index` versus doing a full import of your database.

Index Types

The PingDirectory Server supports several types of indexes to quickly find entries that match search criteria in LDAP operations. The Directory Server uses an attribute's matching rules to normalize its values and uses those values as index keys to a list of matching entry IDs. Entry IDs are integer values that are used to uniquely identify an entry in the backend by means of a set of database index files (`id2entry`, `dn2id`, `dn2uri`, `id2children`, `id2subtree`).

Matching rules are elements defined in the schema that tell the server how to interact with the particular attribute. For example, the uid attribute has an equality matching rule defined in the schema, and thus, has an equality index maintained by the Directory Server. The following table describes the index types:

Table 14: Directory Server Index Types

Index Type	Description
Approximate	Used to efficiently locate entries that match the approximate search filter. It is used to identify which entries are approximately equal to a given assertion. Approximate indexes can only be applied to attributes that have a corresponding approximate matching rule.
Equality	Used to efficiently locate entries that match the equality search filter. It is used to identify which entries are exactly equal to a given assertion. Equality indexes can only be applied to attributes that have a corresponding equality matching rule. An offshoot of the equality index is the filtered index, which uses a defined search filter for a specific attribute. The filtered index can be maintained independently of the equality index for a specific attribute.
Ordering	Used to efficiently locate entries that match the ordering search filter. It is used to identify which entries have a relative order of values for an attribute. Ordering indexes can only be applied to attributes that have a corresponding ordering matching rule.
Presence	Used to efficiently locate entries that match the presence search filter. It is used to identify which entries have at least one value for a specified attribute. There is only one presence index key per attribute.
Substring	Used to efficiently locate entries that match the substring search filter. It is used to identify which entries contain specific substrings to a given assertion. Substring indexes can only be applied to attributes that have a corresponding substring matching rule.

System Indexes

The PingDirectory Server contains a set of system database index files that ensure that the server operates efficiently. These indexes cannot be modified or deleted.

Table 15: System Indexes

Index	Description
dn2id	Allows quick retrieval of DNs. The DN database, or `dn2id`, has one record for each entry. The key is the normalized entry DN and the value is the entry ID.
id2entry	Allows quick retrieval of entries. The `id2entry` database contains the LDAP entries. The database key is the entry ID and the value is the entry contents.
referral	Allows quick retrieval of referrals. The referral database called `dn2uri` contains URIs from referral entries. The key is the DN of the referral entry and the value is that of a labeled URI in the ref attribute for that entry.
id2children	Allows quick retrieval of an entry and its children. The `id2children` database provides a mapping between an entry's unique identifier and the entry unique identifiers of the corresponding entry's children.
id2subtree	Allows quick retrieval of an entry's subtree. The `id2subtree` database provides a mapping between an entry's unique identifier and its unique identifiers in its subtree.

To View the System Indexes

Use the `dbtest` command to view the system and user indexes. The index status indicates if it is a trusted index or not. An index is either trusted or untrusted. An untrusted index requires rebuilding.

```
$ bin/dbtest list-index-status --baseDN dc=example,dc=com --backendID
 userRoot
```

Managing Local DB Indexes

You can modify the local DB indexes to meet your system's requirements using the `dsconfig` tool. If you are using the `dsconfig` tool in interactive command-line mode, you can access the **Local DB Index** menu from the Basic object menu.

To View the List of Local DB Indexes

- Use `dsconfig` with the list-local-db-indexes option to view the default list of indexes.

```
$ bin/dsconfig list-local-db-indexes --backend-name userRoot
```

```
Local DB Index     : Type    : index-type
-------------------:---------:--------------------
aci                : generic : presence
cn                 : generic : equality, substring
ds-entry-unique-id : generic : equality
givenName          : generic : equality, substring
mail               : generic : equality
member             : generic : equality
objectClass        : generic : equality
sn                 : generic : equality, substring
telephoneNumber    : generic : equality
uid                : generic : equality
uniqueMember       : generic : equality
```

To View a Property for All Local DB Indexes

- Use `dsconfig` with the --property option to view a property assigned set for all local DB indexes. Repeat the option for each property that you want to list. In this example, the `prime-index` property specifies if the backend is configured to prime the index at startup.

```
$ bin/dsconfig list-local-db-indexes --property index-entry-limit \
  --property prime-index --backend-name userRoot
```

To View the Configuration Parameters for Local DB Index

- To view the configuration setting of a local DB index, use `dsconfig` with the get-local-db-index-prop option and the `--index-name` and `--backend-name` properties. If you want to view the advanced properties, add the --advanced option to your command.

```
$ bin/dsconfig get-local-db-index-prop --index-name aci \
  --backend-name userRoot
```

To Modify the Configuration of a Local DB Index

You can easily modify an index using the `dsconfig` tool. Any modification or addition of an index requires the indexes to be rebuilt. In general, an index only needs to be built once after it has been added to the configuration.

If you add an index, then import the data using the `import-ldif` tool, then the index will be automatically rebuilt. If you add an index, then add the data using some other method than `import-ldif`, you must rebuild the index using the `rebuild-index` tool.

1. Use `dsconfig` with the set-local-db-index-prop option and the `--index-name` and `--backend-name` properties. In this example, update the `prime-index` property, which loads the index at startup. This command requires the --advanced option to access this property.

```
$ bin/dsconfig set-local-db-index-prop --index-name uid \
  --backend-name userRoot --set prime-index:true
```

2. View the index to verify the change.

```
$ bin/dsconfig get-local-db-index-prop --index-name uid \
  --backend-name userRoot
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in General Tips on Indexes.

```
$ bin/stop-server
```

4. Run the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index uid
```

5. Restart the Directory Server if shutdown.

```
$ bin/start-server
```

To Create a New Local DB Index

1. To create a new local DB index, use `dsconfig` with the --create-local-db-index option and the --index-name, --backend-name, and --set index-type: <value> options.

```
$ bin/dsconfig create-local-db-index \
  --index-name roomNumber --backend-name userRoot \
  --set index-type:equality
```

2. View the index.

```
$ bin/dsconfig get-local-db-index-prop \
  --index-name roomNumber --backend-name userRoot
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in General Tips on Indexes on page 148.

```
$ bin/stop-server
```

4. Rebuild the index using the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index roomNumber
```

5. Restart the Directory Server.

```
$ bin/start-server
```

To Delete a Local DB Index

You can delete an index using the `dsconfig` tool. Check that no plug-in applications are using the index before deleting it. When the index is deleted, the corresponding index database will also be deleted. The disk space is reclaimed once the cleaner threads begin.

- Use `dsconfig` with the `delete-local-db-index` option to remove it from the database.

```
$ bin/dsconfig delete-local-db-index \
  --index-name roomNumber --backend-name userRoot
```

Working with Composite Indexes

The PingDirectory Server composite index can be generated from multiple pieces of information (a combination of multiple filter components, or a combination of filter components and a base DN). A composite index can also be based on only a single piece of information.

To improve searches over a large number of entries, equality composite indexes can be used to combine a mandatory equality filter pattern with an optional base DN pattern to improve the performance of searches in directories with a very large number of entries, and in particular with a very large number of non-leaf entries. Equality composite indexes offer two advantages over existing equality attribute indexes in these types of deployments.

Base DN Pattern - If a directory environment has many branches, but searches are often done that are within specific individual branches, the base DN pattern can be used to make search processing more efficient. The server will only need to search entries within a target branch.

For example, if the directory contains an "`ou=Customers,dc=example,dc=com`" branch, with a separate branches below that for sets of customers, like "`ou=ACME,ou=Customers,dc=example,dc=com`", and "`ou=SHOPCO,ou=Customers,dc=example,dc=com`", a composite index with a filter pattern of "`(sn=?)`" and a base DN pattern of "`ou=?,ou=Customers,dc=example,dc=com`" can be defined. Then a search with a filter of "`(sn=Smith)`" and a base DN of "`ou=ACME,ou=Customers,dc=example,dc=com`" can be used to narrow the search to the Smiths in the ACME branch.

Index Pages - If many entries have the same value for a specific attribute, composite indexes can break large ID sets up across multiple pages, unlike the traditional attribute index. Using the previous example, if a search of the directory returns 50,000 Smiths, the results can be served in blocks of 5,000 IDs. An attribute index will return either one Smith record whose value is a block that contains all 50,000 of the matching entry IDs (if the index isn't exploded), or 50,000 Smith records that each have a value of the matching entry ID (when the index is exploded). The non-exploded form is efficient for searching because all of the entry IDs are returned in a single read, but it's expensive for writing because if a Smith must be added or removed, the entire block of 50,000 entry IDs must be rewritten. The exploded form is efficient for writing (adding or removing a Smith involves just that one entry ID), but it's expensive for searching because it takes 50,000 reads to get all entry IDs for all of the Smiths.

Composite indexes break up the block of entry IDs across multiple pages (a page size of up to 5000). If the directory contains 50,000 Smiths, instead of having to choose between one block of 50,000 IDs or 50,000 blocks of one ID, ten blocks of 5,000 IDs are returned. This improves the efficiency of a read or write across many entries.

There is little performance overhead to the paging mechanism. Use an equality composite index for an attribute that has a lot of entries that have the same value (such as givenName or sn), not for an attribute with very few entries with the same value (such as id or mail). For attributes in which all of the values match a small number of entries, it's better to use an equality attribute index.

When configuring a composite index, define the following properties:

Table 16: Composite Index Properties

Composite Index Properties	Description
index-filter-pattern	Specifies a single-valued filter property used to identify a portion of the index criteria. This can only be specified at the time that the index definition is created and is required.
index-base-dn-pattern	Specifies a single-valued DN property that may indicate that the index should be scoped to a specific subtree or subtree pattern. This can only be specified at the time the index definition is created and is optional.

Working with JSON Indexes

JSON indexing is similar to general attribute indexing. Where an attribute can be indexed several ways and requires a separate database for each index type, there is only a single database for each JSON field that can be used for different JSON filter types. This database primarily behaves like the database for an equality attribute index. Each database entry key is the normalized form for a value for the target JSON field, and the corresponding database entry value is an list of the entry IDs for all entries in which the associated attribute type has a JSON object with that value for the target field. The database is configured with a comparator (based on the data type for the target field) that enables iterating through values in a logical order to facilitate inequality and subInitial searches.

JSON indexes are automatically created when the JSON Field constraint indicates that a JSON field should be indexed. Indexes can be viewed with the following command:

```
$ bin/dbtest list-index-status \
  --backendID userRoot \
  --baseDN dc=example,dc=com
```

The JSON object filter types that can be enhanced through the use of JSON indexes include:

- `equals`. Identifies entries that have a specific value for the target field. This filter type only requires retrieving a single index key. However, depending on the nature of the search filter, the ID list may contain references to entries that don't actually match the filter (such as if the field is a string, and the filter is configured to use case-sensitive matching).
- `equalsAny`. Identifies entries that have any of a specified set of values for the target field. This filter type only requires retrieving the index keys that correspond to the target values in the filter and merging their ID lists.
- `greaterThan/lessThan`. Identifies entries that have at least one value for the target field that is greater or less than (or possibly equal to) a specified value. This index is similar in use to the `containsField` index, except that it only needs to iterate through a subset of the keys. A filter can contain both `greaterThan` and `lessThan` filters to represent a bounded range.
- `substring`. Identifies entries that have a string value for the target field that matches a given `substring`. The index can only be used for substring filters that include a `subInitial` component. In this case, the server iterates through all of the index keys that match the `startsWith` component, and manually compares values against the remainder of the substring assertion.

JSON indexing is available in local database backends backed by Berkeley DB Java Edition. This includes the following:

- Add, delete, modify, and modify DN operations that make changes to JSON objects stored in the server.
- LDIF imports that include JSON objects, including updates to the cache size estimates for the JSON indexes.
- The `rebuild-index` tool and corresponding backend code to make it possible to generate and rebuild indexes for JSON data. It must be possible to build all JSON indexes for all or a specified subset of fields associated with a given attribute type. The `verify-index` tool should also work with JSON indexes to make it possible to check their validity.
- Matching entry count control and debugsearchindex return attribute provide information about relevant JSON index usage.
- Support for monitoring index content and usage.

Note: Exploded indexes and the entry balancing global index do not support JSON objects.

Working with Local DB VLV Indexes

Local DB VLV indexes allow a client to request a subset of results from a sorted list that match a specific search base, scope, and filter. The client can navigate through the list by passing a context back to the server with the virtual list view control. The Local DB VLV index can be used only when the client request contains the virtual list view (VLV) control and the client has been authorized with an ACI with a `targetcontrol` of **2.16.840.1.113730.3.4.9**.

Note: A client request, which includes a virtual list view control, can be successfully processed without a matching Local DB VLV index if the search is completely indexed. This is not an efficient means of using VLV, since the server has to retrieve each entry twice.

To View the List of Local DB VLV Indexes

- Use `dsconfig` with the list-local-db-indexes option to view the default list of indexes. In the example, no VLV indexes are defined.

```
$ bin/dsconfig list-local-db-vlv-indexes --backend-name userRoot
```

To Create a New Local DB VLV Index

1. Use `dsconfig` with the create-local-db-vlv-index option and the --index-name, --backend-name, and --set index-type:(propertyValue) options. If you do not set any property values, the default values are assigned.

```
$ bin/dsconfig create-local-db-vlv-index \
  --index-name givenName --backend-name userRoot --set base-
dn:dc=example,dc=com \
  --set scope:whole-subtree --set filter:"(objectclass=*)" \
  --set sort-order:givenName
```

2. Rebuild the index using the `rebuild-index` tool. You must add the “vlv.” prefix to the index name to rebuild the VLV index. The following command can be run with the server on or offline with the addition of the `--task` and connection options.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

To Modify a VLV Index’s Configuration

1. Use `dsconfig` with the set-local-db-vlv-index-prop option and the `--index-name` and `--backend-name` properties. In this example, update the `base-dn` property.

```
$ bin/dsconfig set-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot --set base-dn:ou=People,dc=example,dc=com
```

2. Rebuild the index using the `rebuild-index` tool. You must add the prefix "vlv." to the index name. The following command can be run with the server on or offline with the addition of the `--task` and connection options.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

To Delete a VLV Index

You can delete a VLV index using the `dsconfig` tool. Check that the index is not being used in any plug-in applications before deleting it.

1. Use `dsconfig` with the delete-local-db-vlv-index option to remove it from the database.

```
$ bin/dsconfig delete-local-db-vlv-index --index-name givenName \
  --backend-name userRoot
```

2. Verify the deletion by trying to view the vlv index.

```
$ bin/dsconfig get-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot
```

Working with Filtered Indexes

The PingDirectory Server filtered index is useful when client search requests consisting of a compound &-filter with individual components matching a large number of entries, potentially greater than the index entry limit, have an intersection of a relatively small number of entries.

For example, assume a database contains several thousand company profiles and each company profile is represented by many entries. The "`(objectClass=company)`" filter matches a small set of entries per company, and therefore may exceed the index entry limit since there are many companies. Also assume that the "`(companyDomain=example.com)`" filter matches many of the entries for the company with domain `example.com` and can also result in an unindexed search. The more narrow filter "`(&(objectClass=company)(companyDomain=example.com))`" also results in an unindexed search but only matches a small number of entries. The filtered index makes it possible to index this compound filter by defining an equality index on the `companyDomain` attribute with a static filter of "`(objectClass=company)`" in the `equality-index-filter` property of the index.

Filtered indexing is primarily useful for cases in which clients frequently issue searches with AND filters that meet the following criteria:

- The AND filter itself matches a relatively small number of entries, but each of the individual components may match a very large number of entries.
- The filter has a dynamic component that does change, and that dynamic component always uses the same attribute.
- The filter has a static component that doesn't change.
- The filter must be narrowed to a base DN, for data structures with many branches, or if indexed attribute values appear in a very large number of entries.

The filtered index can be maintained independently from the equality filter for that attribute. Further, the filtered index will be used only for searches containing the equality component with the associated attribute type ANDed with this filter. When configuring a filtered index, be aware of the `equality-index-filter` and `maintain-equality-index-without-filter` properties of the index.

Once configured and built with the `rebuild-index` tool or `import-ldif`, searches with filters based on the above example will be processed with the index:

```
(&(objectClass=company)(companyDomain=example.com))
(&(objectClass=company)(|(companyDomain=example.com)
(companyDomain=example.org)))
(&(companyDomain=example.com)(objectClass=company))
(&(companyDomain=example.com)(&(objectClass=company)))
(&(companyDomain=example.com)(objectClass=company)(something=else))
(&(companyDomain=example.com)(&(objectClass=company)(something=else)))
(|(&(objectClass=company)(companyDomain=example.com))(&(objectClass=company)
(companyDomain=example.org)))
```

When configuring a filtered index, define the following properties:

Table 17: Filtered Index Properties

Filtered Index Properties	Description
equality-index-filter	Specifies a search filter that may be used in conjunction with an equality component for the associated attribute type. If an equality index filter is defined, then an additional equality index will be maintained for the associated attribute, but only for entries that match the provided filter. Further, the index will be used only for searches containing an equality component with the associated attribute type ANDed with this filter.
maintain-equality-index-without-filter	Specifies whether to maintain a separate equality index for the associated attribute without any filter, in addition to maintaining an index for each equality index filter that is defined. If this is false, then the attribute will not be indexed for equality by itself but only in conjunction with the defined equality index filters.

To Create a Filtered Index

1. Use the `dsconfig` tool to create a filtered index. The following command creates an equality index on the `companyDomain` attribute and maintains an index for the equality filter defined `"(objectclass=company)"`. After you have created the index, you must rebuild the indexes.

```
$ bin/dsconfig create-local-db-index --backend-name "userRoot" \
  --index-name companyDomain --set maintain-equality-index-without-
filter:true \
  --set index-type:equality --set equality-index-
filter:"(objectclass=company)"
```

2. Stop the Directory Server using `bin/stop-server`.
3. Run the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index companyDomain
```

4. Start the Directory Server using `bin/start-server`.

Tuning Indexes

The PingDirectory Server provides several tools to help you optimize your indexes and improve the overall read and write performance for your system. The server now supports an optional expanded index database using an *exploded* format to process high write load operations. To view the current state of the server's indexes and make adjustments to the index databases, the Directory Server automatically generates an Index Summary Statistics Table after each LDIF import or index rebuild. The `dbtest` tool also includes an Index Histogram to determine the key datasize for the indexes. This section provides descriptions of each of these tools.

About the Exploded Index Format

The Directory Server has two index properties on the Backend configuration object that can be used to optimize the server's performance: `index-entry-limit` and `exploded-index-entry-threshold`. The `index-entry-limit` specifies the maximum size of entries kept in an index record, before the server stops maintaining that record, begins scanning the whole database, and runs an expensive unindexed search. Typically, if any index keys have already reached this limit, indexes must be rebuilt before they can be allowed to use a new limit.

The `exploded-index-entry-threshold` property specifies the maximum number of entries that are allowed to match a given index key before that particular index key is stored in a separate database in an expanded (i.e., "exploded") format. An `exploded-index-entry-threshold` value of 0, or any value greater or equal to the `index-entry-limit`, means that the expanded index database will not be used. All keys whose entry count is less than `exploded-index-entry-threshold` continue to be stored in one database in a consolidated format, such that changes to the key require *rewriting* all the entry IDs matching the key. All keys whose entry count is greater than `exploded-index-entry-threshold` and less than the `index-entry-limit` are stored in a *separate* database in an exploded format, such that changes to the key require writing only to the updated entry ID.

For example, as shown in the figure below, the equality index for the `sn` attribute is stored in consolidated format, listing each entry ID for all entries that contain the `sn=smith` attribute. If we enable the `exploded-index-entry-threshold` property by setting its value to 1000 and keep the `index-entry-limit` property to its default limit of 4000, then the expanded index database will be implemented by the server. Any key with an entry count less than 1000 continues to be stored in consolidated format. If a key has an entry count greater than 1000, it will be stored in a separate database where each key is stored with its entry ID individually. The consolidated format is very efficient for read operations because the server can retrieve a row of entry IDs at once, while the exploded format is far more efficient for write operations.

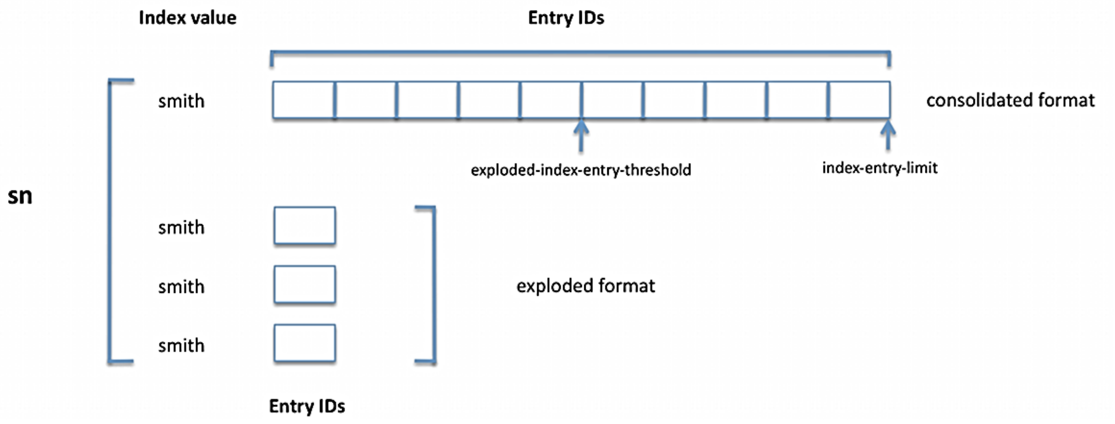


Figure 2: Consolidated and Exploded Index Formats

Note that a large value for the `exploded-index-entry-threshold` can have a big impact on write performance and database growth on disk, since each change to an index key stored in the consolidated format requires that IDs for all entries matching the key be rewritten to the database. For example if the `exploded-index-entry-threshold` is set to 100,000, and the number of entries matching a given key is 50,000, then changes to that key will require writing 200KB to the database since each entry ID is 4 bytes. Add and delete operations are especially impacted since they must update all indexes for an entry. If many indexes have a large `exploded-index-entry-threshold`, this can lead to very low throughput and can cause the database to temporarily grow very large on disk. It is recommended to set this value at or below 10,000, and never set it above 50,000. If any index keys have already reached this threshold, indexes need to be rebuilt before they are allowed to use the new threshold. This is also true for the `index-entry-limit` property: setting a large limit (greater than 10,000) could have a big impact on write performance and database growth on disk.

Monitoring Index Entry Limits

Index keys that have reached their limit require indexes to be rebuilt before they can be allowed to use a new limit. There are several ways to monitor index limits to avoid a potentially costly rebuild. There are cases in which it is acceptable for index keys to exceed the index entry limit. For example, the `objectClass` attribute type should be indexed for equality because the server needs to use it to find all group entries when bringing a backend online, and also because applications frequently need to find entries of a specific type. However, it doesn't make sense for the "top" objectClass key to be indexed, because it appears in every entry in the server.

Choose an index entry limit value that is high enough to ensure that all of the right keys are indexed, but keys that occur too frequently are not. The `verify-index` tool `--listKeysNearestIndexEntryLimt` argument lists a specified number of keys that are closest to the limit without having exceeded it. The index entry limit should be larger than the number of entries matching the largest key to remain indexed, with enough overhead to account for future growth. Use this command regularly to determine if the index entry limit needs to be adjusted.

The `verify-index` tool also provides the `--listKeysExceedingIndexEntryLimit` argument to list all keys for which the value has exceeded the index entry limit and the number of entries in which they appear. If there are keys for which the limit has already exceeded but that need to be maintained, adjust the index entry limit to be higher than the number of entries that contain that key (with additional room for future growth) and run the `rebuild-index` tool (or export to LDIF and re-import).

The server provides other ways to determine if index keys have exceeded, or are close to exceeding, the index entry limit. Some of these include:

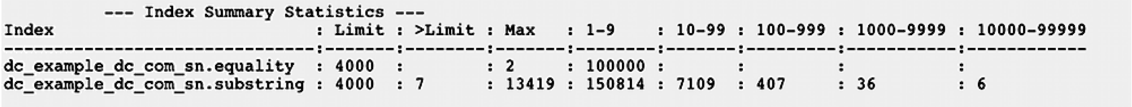
- When performing an LDIF import, the tool includes an "Index Summary Statistics" section that provides usage information for each index, including the number of keys for which the index entry limit has been exceeded, and also the number of keys for which the number of matching entries falls within a number of predefined buckets (such as 1–4 entries, 5–9 entries, 10–99 entries, and 100–999 entries).
- If, during a search operation, the server accesses one or more index keys whose values have exceeded the index entry limit, the access log message for that operation will include an `indexesWithKeysAccessedExceedingEntryLimit` field containing a comma-delimited list of the appropriate indexes. The same access log field may appear in log messages for add, delete, modify, and modify DN operations in which the server wrote to, or tried to write to, at least one index key whose value exceeded the index entry limit.
- If, during a search operation, the server accesses one or more index keys whose values have not yet exceeded the index entry limit but are more than 80 percent to reaching that limit, the access log message for that operation will include an `indexesWithKeysAccessedNearEntryLimit` field containing a comma-delimited list of the appropriate indexes. The same access log field may appear in log messages for add, delete, modify, and modify DN operations in which the server wrote to at least one index key whose value was within 80 percent of the index entry limit.
- If a search operation requests either includes the `debugsearchindex` attribute, or the matching entry count request control with debugging enabled, the debug information will include any indexes accessed that have exceeded the index entry limit, or that are within 80 percent of the configured index entry limit.
- The monitor entry for each configured index includes attributes that provide information about the number of index keys that have been encountered (since the backend was brought online, or since the index entry limit was changed) in a number of different categories. These monitor attributes include:

- `ds-index-exceeded-entry-limit-count-since-db-open` — The number of index keys for which the number of matching entries has crossed the index entry limit due to a write operation.
- `ds-index-unique-keys-near-entry-limit-accessed-by-search-since-db-open` — The number of unique index keys that have been accessed by a search operation for which the number of matching entries is within 80 percent of the index entry limit.
- `ds-index-unique-keys-exceeding-entry-limit-accessed-by-search-since-db-open` — The number of unique index keys that have been accessed by a search operation for which the number of matching entries has exceeded the index entry limit at some point since the index was last built.
- `ds-index-unique-keys-near-entry-limit-accessed-by-write-since-db-open` — The number of unique index keys that have been accessed by a write operation for which the number of matching entries is within 80 percent of the index entry limit.
- `ds-index-unique-keys-exceeding-entry-limit-accessed-by-write-since-db-open` — The number of unique index keys that have been accessed by a write operation for which the number of matching entries has exceeded the index entry limit at some point since the index was last built.

About the Index Summary Statistics Table

The Directory Server now automatically generates an Index Summary Statistics Table, which can be used to determine the optimal configuration for your system's indexes. This table is only generated when the `rebuild-index` tool is run in offline mode. The table is generated after any LDIF import or an index rebuild, and is written to system out and to `logs/tools/rebuild-index-summary.txt`. The table lists the current index entry limit (set by the `index-entry-limit` property on the local DB configuration), the number of keys whose entry count exceeds this limit if any, and the maximum entry count for any key in the index. The table then displays a histogram of the number of keys whose entry falls within a range of values. An example of the Index Summary Statistics table is show below for the `sn.equality` and the `sn.substring` indexes.

The following figure shows that there are seven substrings whose entry counts exceed the `index-entry-limit` of 4000. Six of the substrings are in the 10000-99999 range with the maximum entry count being 13419. By deduction, one more substring must be present in the 1000-9999 range that exceeds the `index-entry-limit` of 4000. These substrings could be expensive for search operations.

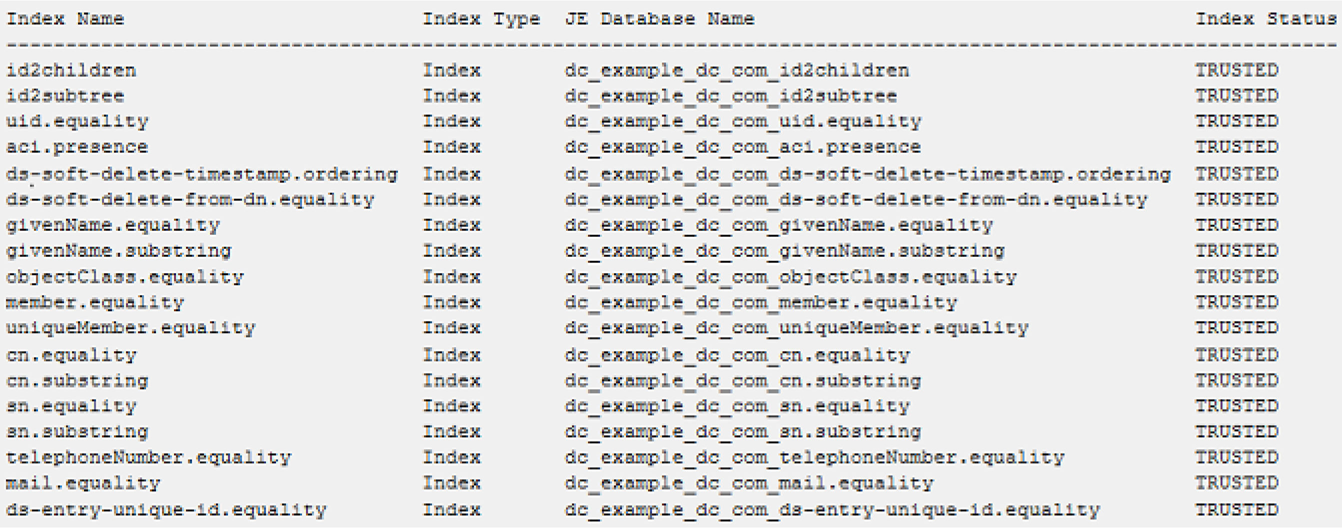


```
          --- Index Summary Statistics ---
Index                          : Limit : >Limit : Max   : 1-9    : 10-99 : 100-999 : 1000-9999 : 10000-99999
-------------------------------:-------:--------:-------:--------:-------:---------:-----------:------------
dc_example_dc_com_sn.equality  : 4000  :        : 2     : 100000 :       :         :           :
dc_example_dc_com_sn.substring : 4000  : 7      : 13419 : 150814 : 7109  : 407     : 36        : 6
```

Figure 3: Example of an Index Summary Statistics Table

About the dbtest Index Status Table

The `dbtest` tool has a `list-all --analyze` option that generates the current status of all of the databases on your system, including all index databases. The table shows the type, entry count (i.e., the number of records in the dtabase), index status (TRUSTED to indicate that the indexes are up-to-date, or UNTRUSTED if the index needs rebuilding), the total data size for each key, the average data size for each key and the maximum data size for each key. Note also that any indexes that are in exploded format are listed on this table.



```
Index Name                          Index Type  JE Database Name                                    Index Status
-----------------------------------------------------------------------------------------------------------------
id2children                         Index       dc_example_dc_com_id2children                       TRUSTED
id2subtree                          Index       dc_example_dc_com_id2subtree                        TRUSTED
uid.equality                        Index       dc_example_dc_com_uid.equality                      TRUSTED
aci.presence                        Index       dc_example_dc_com_aci.presence                      TRUSTED
ds-soft-delete-timestamp.ordering   Index       dc_example_dc_com_ds-soft-delete-timestamp.ordering TRUSTED
ds-soft-delete-from-dn.equality     Index       dc_example_dc_com_ds-soft-delete-from-dn.equality   TRUSTED
givenName.equality                  Index       dc_example_dc_com_givenName.equality                TRUSTED
givenName.substring                 Index       dc_example_dc_com_givenName.substring               TRUSTED
objectClass.equality                Index       dc_example_dc_com_objectClass.equality              TRUSTED
member.equality                     Index       dc_example_dc_com_member.equality                   TRUSTED
uniqueMember.equality               Index       dc_example_dc_com_uniqueMember.equality             TRUSTED
cn.equality                         Index       dc_example_dc_com_cn.equality                       TRUSTED
cn.substring                        Index       dc_example_dc_com_cn.substring                      TRUSTED
sn.equality                         Index       dc_example_dc_com_sn.equality                       TRUSTED
sn.substring                        Index       dc_example_dc_com_sn.substring                      TRUSTED
telephoneNumber.equality            Index       dc_example_dc_com_telephoneNumber.equality          TRUSTED
mail.equality                       Index       dc_example_dc_com_mail.equality                     TRUSTED
ds-entry-unique-id.equality         Index       dc_example_dc_com_ds-entry-unique-id.equality       TRUSTED
```

Figure 4: dbtest Output Including Index Databases

Configuring the Index Properties

By default, the `index-entry-limit` and the `exploded-index-entry-threshold` are set to 4000, which means that the latter property is not enabled due to its having an equal value to that of the `index-entry-limit` property. You can enable the exploded index database by changing the `exploded-index-entry-threshold` property to a value less than the `index-entry-limit` and to a non-zero value using the `dsconfig` tool.

To Configure the Index Properties

Before running the following commands, be aware that you will need to do an index rebuild on the system, which requires a system shutdown, unless the command is run as a task.

1. Run `dsconfig` and set the `index-entry-limit` and the `exploded-index-entry-threshold` to 5000 and 1000, respectively. By default, both values are set to 4000. Changing the `exploded-index-entry-threshold` property to a value less than the `index-entry-limit` and to a non-zero value enables the exploded index database. Remember to include the bind parameters for your system. Once you enter the command, you must confirm that you want to apply the changes to your system.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set index-entry-limit:5000 \
  --set exploded-index-entry-threshold:1000
```

```
One or more configuration property changes require administrative action
or confirmation/notification.  Those properties include:

 * index-entry-limit:  If any index keys have already reached this limit,
 indexes
   must be rebuilt before they will be allowed to use the new limit.
   Setting a large limit (greater than 10,000) could have a big impact on
   write performance and database growth on disk.
 * exploded-index-entry-threshold:  If any index keys have already reached
   this threshold, indexes need to be rebuilt before they are allowed to use
 the
   new threshold. Setting a large limit (greater than 10,000) could have a
 big
   impact on write performance and database growth on disk.

Continue?  Choose 'no' to return to the previous step (yes / no) [yes]: yes
```

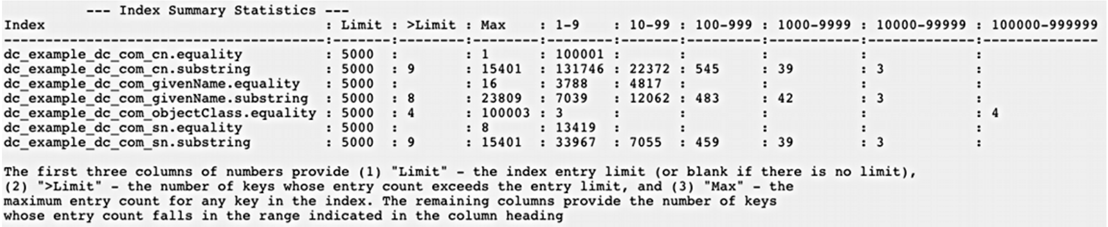
2. Stop the server.

```
$ bin/stop-server
```

3. Rebuild the index.

```
$ bin/rebuild-index --baseDN dc=example,dc=com \
  --index cn --index givenName --index objectClass \
  --index sn --maxThreads 10
```

4. View the Index Summary Statistics table, which is automatically displayed to system out after running the `rebuild-index` command. You can also access the table at `logs/tools/rebuild-index-summary.txt`. Repeat the last three steps to make more adjustments to your indexes.



```
           --- Index Summary Statistics ---
Index                                  : Limit : >Limit : Max    : 1-9    : 10-99 : 100-999 : 1000-9999 : 10000-99999 : 100000-999999
---------------------------------------:-------:--------:--------:--------:-------:---------:-----------:-------------:--------------
dc_example_dc_com_cn.equality          : 5000  :        : 1      : 100001 :       :         :           :             :
dc_example_dc_com_cn.substring         : 5000  : 9      : 15401  : 131746 : 22372 : 545     : 39        : 3           :
dc_example_dc_com_givenName.equality   : 5000  :        : 16     : 3788   : 4817  :         :           :             :
dc_example_dc_com_givenName.substring  : 5000  : 8      : 23809  : 7039   : 12062 : 483     : 42        : 3           :
dc_example_dc_com_objectClass.equality : 5000  : 4      : 100003 : 3      :       :         :           :             : 4
dc_example_dc_com_sn.equality          : 5000  :        : 8      : 13419  :       :         :           :             :
dc_example_dc_com_sn.substring         : 5000  : 9      : 15401  : 33967  : 7055  : 459     : 39        : 3           :

The first three columns of numbers provide (1) "Limit" - the index entry limit (or blank if there is no limit),
(2) ">Limit" - the number of keys whose entry count exceeds the entry limit, and (3) "Max" - the
maximum entry count for any key in the index. The remaining columns provide the number of keys
whose entry count falls in the range indicated in the column heading
```

5. Restart the server.

```
$ bin/start-server
```

Chapter

11

Managing Entries

Topics:

The Directory Server is a fully LDAPv3-compliant server that comes with a comprehensive set of LDAP command-line tools to search, add, modify, and delete entries.

This chapter presents the following topics:

Searching Entries

The Directory Server provides an `ldapsearch` tool to search for entries or attributes within your server. The tool requires the LDAP connection parameters needed to bind to the server, including the `baseDN` option to specify the starting point of the search within the server, and the search scope. The searchScope option determines the depth of the search:

`base` (search only the entry specified)
`one` (search only the children of the entry and not the entry itself)
`sub` (search the entry and its descendents)

The `ldapsearch` tool provides basic functionality as specified by the RFC 2254 but provides additional features that takes advantage of the Directory Server's control mechanisms. For more information, run the `ldapsearch` --help function.

To Search the Root DSE

The Root DSE is a special entry that resides at the root of the directory information tree (DIT). The entry holds operational information about the server and its supported controls. Specifically, the root DSE entry provides information about the supported LDAPv3 controls, SASL mechanisms, password authentication schemes, supported LDAP protocols, additional features, naming contexts, extended operations, and server information.

Note: The Directory Server provides an option to retrieve the Root DSE's operational attributes and add them to the user attribute map of the generated entry. This feature allows client applications that have difficulty handling operational attributes to access the root DSE using the `show-all-attributes` configuration property. Once this property is set, the associated attribute types are re-created and re-registered as user attributes in the schema (in memory, not on disk). Once you set the property, you can use `ldapsearch` without "+" to view the root DSE.

Use the `dsconfig` tool to set the `show-all-attributes` property to TRUE, as follows:

```
$ bin/dsconfig set-root-dse-backend-prop --set show-all-attributes:true
```

- Use `ldapsearch` to view the root DSE entry on the Directory Server. Be sure you include the "+" to display the operational attributes in the entry.

```
$ bin/ldapsearch --baseDN "" --searchScope base "(objectclass=*)" "+"
```

To Search All Entries in the Directory Server

- Use `ldapsearch` to search all entries in the Directory Server. The filter `"(objectclass=*)"` matches all entries. If the --searchScope option is not specified, the command defaults to a search scope of sub:

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --searchScope sub "(objectclass=*)"
```

To Search for an Access Control Instruction

- Use `ldapsearch` to search the `dc=example,dc=com` base DN entry. The filter `"(aci=*)"` matches all `aci` attributes under the base DN, and the `aci` attribute is specified so that only it is returned. The `cn=Directory Manager` bind DN has the privileges to view an ACI.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(aci=*)" aci
```

```
dn: dc=example,dc=com
aci: (targetattr!="userPassword")
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Search for the Schema

- Use `ldapsearch` to search the `cn=schema` entry. The base DN is specified as `cn=schema`, and the filter `"(objectclass=*)"` matches all entries. The command uses a special attribute "+" to return all operational attributes:

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base "(objectclass=*)" "+"
```

To Search for a Single Entry using Base Scope and Base DN

- Use `ldapsearch` to search for a single entry by specifying the base scope and DN:

```
$ bin/ldapsearch --baseDN uid=user.14,ou=People,dc=example,dc=com \
  --searchScope base "(objectclass=*)"
```

To Search for a Single Entry Using the Search Filter

- Search for a single entry by specifying the sub scope and a search filter that describes a single entry:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub "(uid=user.14)"
```

To Search for All Immediate Children for Restricted Return Values

- Search for all immediate children of `ou=People,dc=example,dc=com`. The attributes returned are restricted to `sn` and `givenName`. The special attribute "+" returns all operational attributes:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope one '(objectclass=*)' sn givenName "+"
```

To Search for All Children of an Entry in Sorted Order

- Search for all children of the `ou=People,dc=example,dc=com` subtree. The resulting entries are sorted by the server in ascending order by `sn` and then in descending order by `givenName`:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub --sortOrder sn,-givenName '(objectclass=*)'
```

To Limit the Number of Returned Search Entries and Search Time

- Search for a subset of the entries in the `ou=People,dc=example,dc=com` subtree by specifying a compound filter. No more than 200 entries will be returned and the server will spend no more than 5 seconds processing the request. Returned attributes are restricted to a few operational attributes:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub --sizeLimit 200 --timeLimit 5 \
  "(&(sn<=Doe)(employeeNumber<=1000))" ds-entry-unique-id entryUUID
```

To Get Information about How Indexes are used in a Search Operation

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Information about the can be obtained about the server's use of indexes in the course of processing a search operation. The following are two basic ways to accomplish this.

Issue a search request with the desired base DN, scope, and filter, and request that the server return the special `debugsearchindex` attribute. Users will need to be granted access to the `debugsearchindex` operational attribute and the `cn=debugsearch` portion of the DIT with the following command:

```
$ bin/dsconfig set-access-control-handler-prop \
 --add "global-aci:(targetattr=\"debugsearchindex\")(target=\"ldap:///
cn=debugsearch\")
    (version 3.0; acl \"Allow members of the Index Debugging Users group
    to request the debugsearchindex operational attribute \"; allow
    (read,search,compare) groupdn=\"ldap:///cn=Index Debugging
    Users,ou=Groups,dc=example,dc=com\";)"
```

To issue a search request for the server return the special `debugsearchindex` attribute, use the `ldapsearch` command such as:

```
$ bin/ldapsearch --hostname ds.example.com \
 --port 389 --bindDN uid=admin,dc=example,dc=com \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --searchScope sub "(&(givenName=John)(sn=Doe))" debugsearchindex
 dn: cn=debugsearch
 debugsearchindex: 0.040 ms - Beginning index processing for search
     request with base DN 'dc=example,dc=com', scope wholeSubtree,
     and filter (&(givenName=John)(sn=Doe)).
 debugsearchindex: 0.067 ms - Unable to optimize the AND filter
     beyond what the client already provided.
 debugsearchindex: 0.834 ms - Candidate set obtained for single-key
     filter (givenName=John) from index dc_example_dc_com_givenName.equality.
     Candidate set:  CandidateSet(isDefined=true, isExploded=false,
     isResolved=true, size=2, originalFilter=(givenName=John),
     remainingFilter=null, matchingEntryCountType=UNEXAMINED_COUNT)
 debugsearchindex: 0.030 ms - Final candidate set for filter (givenName=John)
     obtained from an unexploded index key in
 dc_example_dc_com_givenName.equality.
     Since the scope of the search includes the entire entry container, there
 is
     no need to attempt to further pare down the results based on the search
 scope.
     Candidate set:  CandidateSet(isDefined=true, isExploded=false,
 isResolved=true,
     size=2, originalFilter=(givenName=John), remainingFilter=null,
     matchingEntryCountType=UNEXAMINED_COUNT)
 debugsearchindex: 0.020 ms - Short-circuiting index processing for AND filter
     (&(givenName=John)(sn=Doe)) after evaluating single-key component
     (givenName=John) because the current ID set size of 2 is within the
      short-circuit threshold of 5.
 debugsearchindex: 0.030 ms - Obtained a candidate set of size 2 for AND
 filter
     (&(givenName=John)(sn=Doe)) with remaining filter (sn=Doe).  Even though
     there is still more of the filter to evaluate, the current candidate set
 is
     within the short-circuit threshold of 5, so no additional index
 processing will
     be performed to try to pare down the results based on the remaining
 filter or the
     search scope.  Candidate set:  CandidateSet(isDefined=true,
 isExploded=false,
     isResolved=true, size=2, originalFilter=(&(givenName=John)(sn=Doe)),
     remainingFilter=(sn=Doe), matchingEntryCountType=UPPER_BOUND)
 debugsearchindex: 0.016 ms - Completed all index processing.  Candidate set:
     CandidateSet(isDefined=true, isExploded=false, isResolved=true, size=2,
     originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
     matchingEntryCountType=UPPER_BOUND)
```

The second way would be to issue a search request with the desired base DN, scope, and filter, and include the matching entry count request control with the debug option set to `true`. To do this with `ldapsearch`, use a command such as:

```
$ bin/ldapsearch --hostname ds.example.com --port 389 \
 --bindDN uid=admin,dc=example,dc=com --bindPassword password \
 --baseDN dc=example,dc=com \
 --searchScope sub --matchingEntryCountControl examineCount=0:debug
 "(&(givenName=John)(sn=Doe))"
 Upper Bound on Matching Entry Count:  2
 Matching Entry Count Debug Messages:
 * naw-desktop:1389 - 0.104 ms - Beginning index processing for search request
 with
    base DN 'dc=example,dc=com', scope wholeSubtree, and filter
 (&(givenName=John)(sn=Doe)).
 * naw-desktop:1389 - 0.105 ms - Unable to optimize the AND filter beyond what
 the client already provided.
 * naw-desktop:1389 - 0.614 ms - Candidate set obtained for single-key filter
 (givenName=John) from index
    dc_example_dc_com_givenName.equality.  Candidate set:
    CandidateSet(isDefined=true, isExploded=false, isResolved=true,
    size=2, originalFilter=(givenName=John), remainingFilter=null,
 matchingEntryCountType=UNEXAMINED_COUNT)
 * naw-desktop:1389 - 0.090 ms - Final candidate set for filter
    (givenName=John) obtained from an unexploded index key in
    dc_example_dc_com_givenName.equality.  Since the scope of the search
    includes the entire entry container, there is no need
    to attempt to further pare down the results based on the search scope.
    Candidate set:  CandidateSet(isDefined=true, isExploded=false,
 isResolved=true,
    size=2, originalFilter=(givenName=John), remainingFilter=null,
 matchingEntryCountType=UNEXAMINED_COUNT)
 * naw-desktop:1389 - 0.045 ms - Short-circuiting index processing for AND
 filter
    (&(givenName=John)(sn=Doe)) after evaluating single-key component
    (givenName=John) because the current ID set size of 2 is within the short-
circuit threshold of 5.
 * naw-desktop:1389 - 0.111 ms - Obtained a candidate set of size 2 for AND
 filter
    (&(givenName=John)(sn=Doe)) with remaining filter (sn=Doe).  Even though
 there is
    still more of the filter to evaluate, the current candidate set is within
    the short-circuit threshold of 5, so no additional index processing will
 be performed
    to try to pare down the results based on the remaining filter or the
 search scope.
    Candidate set:  CandidateSet(isDefined=true, isExploded=false,
 isResolved=true, size=2,
    originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
    matchingEntryCountType=UPPER_BOUND)
 * naw-desktop:1389 - 0.040 ms - Completed all index processing.  Candidate
 set:
    CandidateSet(isDefined=true, isExploded=false, isResolved=true, size=2,
    originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
    matchingEntryCountType=UPPER_BOUND)
 * naw-desktop:1389 - The search is partially indexed
 (candidatesAreInScope=true,
    unindexedFilterPortion=(sn=Doe))
 * naw-desktop:1389 - Constructing an UPPER_BOUND response with a count of 2
```

Working with the Matching Entry Count Control

The ldapsearch command can be used with the --matchingEntryCountControl option to determine the count of entries that match a search filter. Users will need to be granted access to this control by its OID 1.3.6.1.4.1.30221.2.5.36 with the following command:

```
$ bin/dsconfig set-access-control-handler-prop \
 --add "global-aci:(targetcontrol=\"1.3.6.1.4.1.30221.2.5.36\")
   (version 3.0; acl \"Allow members of the Index Debugging Users group to
   use the matching entry count request control \"; allow (read)
   groupdn=\"ldap:///cn=Index Debugging Users,ou=Groups,dc=example,dc=com\";)"
```

An examineCount control can be used for searches that are at least partially indexed, in order to determine whether to return an examined count, an unexamined count, or an upper bound count. The factors that determine what is returned are:

- A search is fully indexed if indexes can be used to identify the entry IDs for all entries that match the filter without ambiguity. Indexes can also be used to make sure that all of those candidates are within the scope of the search.
- A search is partially indexed if indexes can be used to identify the entry IDs for all entries that match the search criteria, but the candidate list may potentially also include entries that either don't match the filter or are outside the scope of the search.
- A search is unindexed if it is not possible to retrieve a candidate list based on either the filter or the search scope.
- An unexamined count is a count of the exact number of entries that match the search criteria, only through the use of index processing.
- An examined count is the same as an unexamined count, except that all of the candidate entries are examined to determine whether they would have been returned to the client. An examined count may be less than an unexamined count if the set of matching entries includes those that would be removed by access control evaluation, or special entries like LDAP sub-entries, replication conflict entries, or soft-deleted entries.
- An upper bound count is the maximum number of entries that match the criteria, but indicates that the server could not determine exactly how many matching entries there were without examining each candidate, which it did not do.
- If a search is fully indexed, the result is an examined count or an unexamined count. If alwaysExamine is true and examineCount is greater than or equal to the number of candidates, the result is an examined count. If alwaysExamine is false, or if the number of candidates exceeds examineCount, the result is an unexamined count.
- If a search is partially indexed, the result is either an examined count or an upper bound count. The alwaysExamine flag isn't relevant in this case. If examineCount is greater than or equal to the number of candidates, the result is an examined count. If not, the result is an upper bound count.
- If a search is unindexed, the result is either an examined count or an unknown count. If allowUnindexed is true, the unindexed search is processed, which can be very expensive. Instead of getting the matching entries back, the examined count is returned. If allowUnindexed is false, an unknown count is returned. If allowUnindexed is true, the requester needs to have the unindexed-search privilege to get the exact count.

The following is a sample ldapsearch with the --matchingEntryCountControl option:

```
$./ldapsearch \
  --bindDN "cn=directory manager" \
  --bindPassword password \
  --baseDN dc=example,dc=com \
  --matchingEntryCountControl
 examineCount=100:alwaysExamine:allowUnindexed:debug \
  "(objectclass=*)"
```

Adding Entries

Depending on the number of entries that you want to add to your Directory Server, you can use the `ldapmodify` tool for small additions. The `ldapmodify` tool provides two methods for adding a single entry: using a LDIF file or from the command line. The attributes must conform to your schema and contain the required object classes.

Adding requests with the `ignore-no-user-modification` control enable a client to include attributes that are not normally allowed from external sources. For example, the `userPassword` attribute is a user-modifiable attribute. An add request with the `ignore-no-user-modification` control allows a one-time exception to the password policy, even if the requesting client does not have the `bypass-pw-policy` privilege. This exception enables specifying pre-encoded passwords.

Note: When adding an entry, the server can ensure that the entry's RDN is unique and does not contain any sensitive information by replacing the provided entry's RDN with the server-generated entryUUID value. An LDAP client written with the LDAP SDK for Java can use the `NameWithEntryUUIDRequestControl` to explicitly indicate which add requests should be named in this way, or the `ldapmodify` tool with the `--nameWithEntryUUID` argument. Also, the `auto-name-with-entry-uuid-connection-criteria` and `auto-name-with-entry-uuid-request-criteria` global configuration properties can be used to identify which add requests should be automatically named this way.

The uniqueness request control can also be used with `ldapmodify` for enforcing uniqueness on a per-request basis. Provide at least one of the `uniquenessAttribute` or `uniquenessFilter` arguments with the request. For more information about this control, see the LDAP SDK documentation and the `com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl` class for using the control.

To Add an Entry Using an LDIF File

1. Open a text editor and create an entry that conforms with your schema. For example, add the following entry in the file and save the file as `add-user.ldif`. For the `userPassword` attribute, enter the cleartext password. The Directory Server encrypts the password and stores its encrypted value in the server. Make sure that the LDIF file has limited read permissions for only authorized administrators.

```
dn: uid=user.2000,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Toby Hall$73600 Mash Street$Cincinnati, OH 50563 postalCode:
 50563
description: This is the description for Toby Hall.
uid: user.2000
userPassword: wordsmith employeeNumber: 2000
initials: TBH
givenName: Toby
pager: +1 596 232 3321
mobile: +1 039 311 9878
cn: Toby Hall
sn: Hall
telephoneNumber: +1 097 678 9688
street: 73600 Mash Street
homePhone: +1 214 233 8484
l: Cincinnati
mail: user.2000@maildomain.net
st: OH
```

2. Use the `ldapmodify` tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. If the command is successful, you will see generated success messages with the "#" symbol.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

```
# Processing ADD request for uid=user.2000,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2000,ou=People,dc=example,dc=com
```

To Add an Entry Using the Changetype LDIF Directive

RFC 2849 specifies LDIF directives that can be used within your LDIF files. The most commonly used directive is `changetype`, which follows the `dn:` directive and defines the operation on the entry. The main advantage of using this method in an LDIF file is that you can combine adds and modifies in one file.

1. Open a text editor and create an entry that conforms with your schema. For example, add the following entry in the file and save the file as `add-user2.ldif`. Note the use of the `changetype` directive in the second line.

```
dn: uid=user.2001,ou=People,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Seely Dorm$100 Apple Street$Cincinnati, OH 50563
postalCode: 50563
description: This is the description for Seely Dorm.
uid: user.2001
userPassword: pleasantry
employeeNumber: 2001
initials: SPD
givenName: Seely pager: +1 596 665 3344
mobile: +1 039 686 4949
cn: Seely Dorm
sn: Dorm
telephoneNumber: +1 097 257 7542
street: 100 Apple Street
homePhone: +1 214 521 4883
l: Cincinnati
mail: user.2001@maildomain.net
st: OH
```

2. Use the `ldapmodify` tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. In this example, you do not need to use the --defaultAdd or its shortform -a option with the command.

```
$ bin/ldapmodify --filename add-user2.ldif
```

To Add Multiple Entries in a Single File

You can have multiple entries in your LDIF file by simply separating each DN and its entry with a blank line from the next entry.

1. Open a text editor and create a couple of entries that conform to your schema. For example, add the following entries in the file and save the file as `add-user3.ldif`. Separate each entry with a blank line.

```
dn: uid=user.2003,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass:
organizationalPerson
objectClass: inetOrgPerson
...(similar attributes to previous examples)...

dn: uid=user.2004,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

```
...(similar attributes to previous examples)...
```

2. Use the `ldapmodify` tool to add the entries specified in the LDIF file. In this example, we use the short form arguments for the `ldapmodify` tool.

   The -h option specifies the host name, the -p option specifies the LDAP listener port, -D specifies the bind DN, -w specifies the bind DN password, -a specifies that entries that omit a changetype will be treated as add operations, and -f specifies the path to the input file. If the operation is successful, you will see commented messages (those begining with "#") for each addition.

```
$ bin/ldapmodify -h server.example.com -p 389 \
  -D "cn=admin,dc=example,dc=com" -w password -a -f add-user3.ldif
```

```
# Processing ADD request for uid=user.2003,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2003,ou=People,dc=example,dc=com
# Processing ADD request for uid=user.2004,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2004,ou=People,dc=example,dc=com
```

Deleting Entries Using ldapdelete

You can delete an entry using the `ldapdelete` tool. You should ensure that there are no child entries below the entry as that could create an orphaned entry. Also, make sure that you have properly backed up your system prior to removing any entries.

To Delete an Entry Using ldapdelete

Use `ldapdelete` to delete an entry. The following example deletes the `uid=user.14` entry.

```
$ bin/ldapdelete uid=user.14,ou=People,dc=example,dc=com
```

To Delete Multiple Entries Using an LDIF File

1. You can generate a file of DNs that you would like to delete from the Directory Server.

   The following command searches for all entries in the `ou=Accounting` branch and returns the DNs of the subentries.

```
$ bin/dump-dns -D "cn=admin,dc=example,dc=com" -w password --baseDN \
                        "ou=Accounting,ou=People,dc=example,dc=com" --
outputFile /usr/local/entry_dns.txt
```

2. Run the `ldapdelete` command with the file to delete the entries. The command uses the --continueError option, which will continue deleting through the whole list even if an error is encountered for a DN entry.

```
$ bin/ldapdelete --filename /usr/local/entry_dns.txt --continueError
```

Deleting Entries Using ldapmodify

You can use the LDIF changetype directive to delete an entry from the Directory Server using the `ldapmodify` tool. You can only delete leaf entries.

To Delete an Entry Using ldapmodify

- From the command line, use the `ldapmodify` tool with the changetype:delete directive. Enter the DN, press **Enter** to go to the next line, then enter the changetype directive. Press **Control-D** twice to enter the EOF sequence (UNIX) or **Control-Z** (Windows).

```
$ bin/ldapmodify --hostname server1.example.com -port 389 --bindDN
 "uid=admin,dc=example,dc=com" --bindPassword password
dn:uid=user.14,ou=People,dc=example,dc=com
```

```
changetype: delete
```

Modifying Entries Using ldapmodify

You can use the `ldapmodify` tool to modify entries from the command line or by using an LDIF file that has the `changetype:modify` directive and value. If you have more than one change, you can separate them using the - (dash) symbol.

To Modify an Attribute from the Command Line

1. Use the `ldapsearch` tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the `ldapmodify` command to change attributes from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be changed using the replace directive. In this example, we change the telephone number of a specific user entry. When you are done typing, you can press **CTRL-D** (Unix EOF escape sequence) twice or **CTRL-Z** (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

To Modify Multiple Attributes in an Entry from the Command Line

1. Use the `ldapsearch` tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the `ldapmodify` command to change attributes from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be changes using the add and replace directive.

   In this example, we add the `postOfficeBox` attribute, change the mobile and telephone numbers of a specific user entry. The `postOfficeBox` attribute must be present in your schema to allow the addition. The three changes are separated by a dash ("-"). When you are done typing, you can press **CTRL-D** (Unix EOF escape sequence) twice or **CTRL-Z** (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postOfficeBox
postOfficeBox: 111
-
replace: mobile
mobile: +1 039 831 3737
-
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

To Add an Attribute from the Command Line

- Use the `ldapmodify` command from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be added using the add directive. In this example, we add another value for the `cn` attribute, which is multi-valued. When you are done typing, you can press **CTRL-D** (UNIX EOF escape sequence) twice to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: cn
cn: Sally Tea Tree
```

An error could occur if the attribute is single-valued, if the value already exists, if the value does not meet the proper syntax, or if the value does not meet the entry's objectclass requirements. Also, make sure there are no trailing spaces after the attribute value.

To Add an Attribute Using the Language Subtype

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

```
$ bin/ldapmodify -h server.example.com -p 389 -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postalAddress; lang-ko
postalAddress; lang-ko:Byung-soon Kim$2020-14 Seoul
```

To Add an Attribute Using the Binary Subtype

The Directory Server provides support for attributes using binary subtypes, which are typically used for certificates or JPEG images that could be stored in an entry. The operation must specifically match the subtype for successful operation. The version directive with a value of "1" must be used for binary subtypes. Typical binary attribute types are `userCertificate` and `jpegPhoto`.

- Use the `ldapmodify` command to add an attribute with a binary subtype. The attribute points to the file path of the certificate.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
version: 1
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate;binary
userCertificate;binary:<file:///path/to/cert
```

To Delete an Attribute

Use `ldapmodify` with the LDIF delete directive to delete an attribute.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: employeeNumber
```

To Delete One Value from an Attribute with Multiple Values

You can use the LDIF delete directive to delete a specific attribute value from an attribute. For this example, assuming you have multiple values of `cn` in an entry (e.g., `cn: Sally Tree`, `cn: Sally Tea Tree`).

- Use `ldapmodify` to delete a specific attribute of a multi-valued pair, then specify the attribute pair that you want to delete. In this example, we keep `cn:Sally Tree` and delete the `cn: Sally Tea Tree`.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: cn
cn: Sally Tea Tree
```

To Rename an Entry

Renaming an entry involves changing the relative distinguished name (RDN) of an entry. You cannot rename a RDN if it has children entries as this violates the LDAP protocol.

- Use the `ldapmodify` tool to rename an entry. The following command changes `uid=user.14` to `uid=user.2014` and uses the changetype, `newrdn`, and `deleteoldrdn` directives.

```
$ bin/ldapmodify
dn: uid=user.14,ou=People,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 1
```

To Move an Entry Within a Directory Server

You can use `ldapmodify` to move an entry from one base DN to another base DN. Before running the `ldapmodify` command, you must assign access control instructions (ACIs) on the parent entries. The source parent entry must have an ACI that allows export operations: `allow(export)`. The target parent entry must have an ACI that allows import operations: `allow(import)`. For more information on access control instructions, see Working with Access Control.

- Use the `ldapmodify` command to move an entry from the Contractor branch to the `ou=People` branch.

```
$ bin/ldapmodify
dn: uid=user.14,ou=contractors,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 0
newsuperior: ou=People,dc=example,dc=com
```

To Move an Entry from One Machine to Another

The Directory Server provides a tool, `move-subtree`, to move a subtree or one entry on one machine to another. The subtree or entry must exist on the source server and must not be present on the target server. The source server must also support the 'real attributes only' request control. The target server must support the Ignore NO-USER-MODIFICATION request control.

Note: The `move-subtree` tool moves a subtree or multiple entries from one machine to another. The tool does not copy the entries. Once the entries are moved, they are no longer present on the source server.

- Use the `move-subtree` tool to move an entry (e.g., `uid=test.user,ou=People,dc=example,dc=com`) from the source host to the target host.

```
$ bin/move-subtree --sourceHost source.example.com --sourcePort 389 \
  --sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password
 \
  --targetHost target.example.com --targetPort 389 \
  --targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password
 \
  --entryDN uid=test.user,ou=People,dc=example,dc=com
```

To Move Multiple Entries from One Machine to Another

The `move-subtree` tool provides the ability to move multiple entries listed in a DN file from one machine to another. Empty lines and lines beginning with the octothorpe character (#) will be ignored. Entry DNs may optionally be prefixed with `dn:` , but long DNs cannot be wrapped across multiple lines.

1. Open a text file, enter a list of DNs, one DN per line, and then save the file. You can also use the `ldapsearch` command with the special character "1.1" to create a file containing a list of DNs that you want to move. The following example searches for all entries that match (`department=Engineering`) and returns only the DNs that match the criteria. The results are re-directed to an output file, `test-dns.ldif`:

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com \
     --searchScope sub "(department=Engineering)" "1.1" > test-dns.ldif
   ```
2. Run the `move-subtree` tool with the --entryDNFile option to specify the file of DNs that will be moved from one machine to another.

   ```
   $ bin/move-subtree --sourceHost source.example.com --sourcePort 389 \
     --sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password
    \
     --targetHost target.example.com --targetPort 389 \
     --targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password
    \
     --entryDNFile /path/to/file/test-dns.ldif
   ```
3. If an error occurs with one of the DNs in the file, the output message shows the error. The `move-subtree` tool will continuing processing the remaining DNs in the file.

   ```
   An error occurred while communicating with the target server: The entry
   uid=user.2,ou=People,dc=example,dc=com cannot be added because an entry with
    that name
   already exists
   Entry uid=user.3,ou=People,dc=example,dc=com was successfully moved from
   source.example.com:389 to target.example.com:389
   Entry uid=user.4,ou=People,dc=example,dc=com was successfully moved from
   source.example.com:389 to target.example.com:389
   ```

Working with the Parallel-Update Tool

The PingDirectory Server provides a `parallel-update` tool, which reads change information (add, delete, modify, and modify DN) from an LDIF file and applies the changes in parallel. This tool is a multi-threaded version of the `ldapmodify` tool that is designed to process a large number of changes as quickly as possible.

The `parallel-update` tool provides logic to prevent conflicts resulting from concurrent operations targeting the same entry or concurrent operations involving hierarchically-dependent entries (for example, modifying an entry after it has been added, or adding a child after its parent). The tool also has a retry capability that can help ensure that operations are ultimately successful even when interdependent operations are not present in the correct order in the LDIF file (for example, the change to add a parent entry is provided later in the LDIF file than a change to add a child entry).

After the tool has applied the changes and reaches the end of the LDIF file, it automatically displays the update statistics described in the following table

Table 18: Parallel-Update Tool Result Statistics

Processing Statistic	Description
Attempts	Number of update attempts
Successes	Number of successful update attempts
Rejects	Number of rejected updates
ToRetry	Number of updates that will be retried
AvgOps/S	Average operations per second
RctOps/S	Recent operations per second. Total number of operations from the last interval of change updates.
AvgDurMS	Average duration in milliseconds
RctDurMS	Recent duration in milliseconds. Total duration from the last interval of change updates.

To Run the Parallel-Update Tool

1. Create an LDIF file with your changes. The third change will generate a rejected entry because its `userPassword` attribute contains an encoded value, which is not allowed.

```
dn:uid=user.2,ou=People,dc=example,dc=com
changetype: delete

dn:uid=user.99,ou=People,dc=example,dc=com
changetype: moddn
newrdn: uid=user.100
deleteoldrdn: 1

dn:uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==

dn:uid=user.100,ou=People,dc=example,dc=com
changetype: modify
replace: st
st: TX
-
replace: employeeNumber
employeeNumber: 100
```

2. Use `parallel-update` to apply the changes in the LDIF file to a target server. In this example, we use ten concurrent threads. The optimal number of threads depends on your underlying system. The --ldifFile and --rejectFile options are also required.

```
$ bin/parallel-update --hostname 127.0.0.1 \
  --ldifFile changes.ldif --rejectFile reject.ldif --numThreads 10
```

```
Reached the end of the LDIF file
Attempts Successes Rejects ToRetry AvgOps/S RctOps/S AvgDurMS RctDurMS
-------- --------- ------- ------- -------- -------- -------- ---------
```

```
       4         3       1       0        3        3        26       26
All processing complete Attempted 4 operations in 1 seconds
```

3. View the rejects file for any failed updates.

```
# ResultCode=53, Diagnostic Message=Pre-encoded passwords are not allowed
 for
# the password attribute userPassword
dn: uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==
```

Working with the Watch-Entry Tool

The PingDirectory Server provides a `watch-entry` tool, which demonstrates replication or synchronization latency by watching an LDAP entry for changes. If the entry changes, the background of modified attributes will temporarily be red. Attributes can also be directly modified with this tool as well.

To Run the Watch-Entry Tool

- Perform the following to connect to `server.example.com` as `uid=admin,dc=example,dc=com` and watch entry `uid=kate,ou=people,dc=example,dc=com` for changes:

```
$ bin/watch-entry --hostname server.example.com --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --entryDN uid=user,ou=people,dc=example,dc=com
```

Working with LDAP Transactions

The PingDirectory Server provides support for *batched transactions*, which are processed all at once at commit time. Applications developed to perform batched transactions should include as few operations in the transaction as possible. The changes are not actually processed until the commit request is received. Therefore, the client cannot know whether the changes will be successful until commit time. If any of the operations fail, then the entire set of operations fails.

Batched transactions are write operations (add, delete, modify, modify DN, and password modify) that are processed as a single atomic unit when the commit request is received. If an abort request is received or an error occurs during the commit request, the changes are rolled back. The batched transaction mechanism supports the standard LDAP transaction implementation based on RFC 5805. It is not currently possible to process a transaction that requires changes to be processed across multiple servers or multiple Directory Server backends.

Directory servers may limit the set of controls that are available for use in requests that are part of a transaction. RFC 5805 section 4 indicates that the following controls may be used in conjunction with the transaction specification request control: assertion request control, manageDsaIT request control, pre-read request control, and post-read request control. The proxied authorization v1 and v2 controls cannot be included in requests that are part of a transaction, but they can be included in the start transaction request to indicate that all operations within the transaction should be processed with the specified authorization identity.

The PingDirectory Server supports the following additional controls in conjunction with operations included in a transaction: account usable request control, hard delete request control, intermediate client request control, password policy request control, replication repair request control, soft delete request control, soft deleted entry access request control, subtree delete request control, and undelete request control.

To Request a Batched Transaction Using ldapmodify

1. Use the `ldapmodify` tool's --useTransaction option. It provides a mechanism for processing multiple operations as part of a single batched transaction. Create a batch text file with the changes that you want to apply as a single atomic unit:

   ```
   dn:uid=user.3,ou=People,dc=example,dc=com
   changetype: delete
   dn:uid=user.1,ou=People,dc=example,dc=com
   changetype: modify
   replace: pager
   pager: +1 383 288 1090
   ```

2. Use `ldapmodify` with the --useTransaction and --filename options to run the batched transaction.

   ```
   $ bin/ldapmodify --useTransaction --filename test.ldif
   ```

   ```
   #Successfully created a transaction with transaction ID 400
   #Processing DELETE request for uid=user.3,ou=People,dc=example,dc=com
   #DELETE operation successful for DN uid=user.3,ou=People,dc=example,dc=com
   #This operation will be processed as part of transaction 400
   #Processing MODIFY request for uid=user.1,ou=People,dc=example,dc=com
   #MODIFY operation successful for DN uid=user.1,ou=People,dc=example,dc=com
   #This operation will be processed as part of transaction 400
   #Successfully committed transaction 400
   ```

Chapter 12

Working with Virtual Attributes

Topics:

The PingDirectory Server provides mechanisms to support virtual attributes in an entry. Virtual attributes are abstract, dynamically generated attributes that are invoked through an LDAP operation, such as `ldapsearch`, but are not stored in the Directory Server backend. While most virtual attributes are operational attributes, providing processing-related information that the server requires, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your directory server requirements.

This chapter presents the following topics:

Overview of Virtual Attributes

The PingDirectory Server allows its entries to hold virtual attributes. Virtual attributes are dynamically generated attributes that are invoked through an LDAP operation, such as `ldapsearch`, but are not stored in the Directory Server backend. Most virtual attributes are operational attributes, providing processing-related information that the server requires. However, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your requirements.

Viewing the List of Default Virtual Attributes

The Directory Server has a default set of virtual attributes that can be viewed using the `dsconfig` tool. Some virtual attributes are enabled by default and are useful for most applications. You can easily enable or disable each virtual attribute using the `dsconfig` tool.

The default set of virtual attributes are described in the table below. You can enable or disable these attributes using the `dsconfig` tool.

Table 19: Virtual Attrbutes

Virtual Attributes	Description
ds-entry-checksum	Generates a simple checksum of an entry’s contents, which can be used with an LDAP assertion control to ensure that the entry has not been modified since it was last retrieved.
ds-instance-name	Generates the name of the Directory Server instance from which the associated entry was read. This virtual attribute can be useful in load-balancing environments to determine the instance from which an entry was retrieved.
entryDN	Generates an `entryDN` operational attribute in an entry that holds a normalized copy of the entry's current distinguished name (DN). Clients can use this attribute in search filters.
hasSubordinates	Creates an operational attribute that has a value of TRUE if the entry has subordinate entries.
isMemberOf	Generates an `isMemberOf` operational attribute that contains the DNs of the groups in which the user is a member.
numSubordinates	Generates an operational attribute that returns the number of child entries. While there is no cost if this operational attribute is enabled, there could be a performance cost if it is requested. Note that this operational attribute only returns the number of immediate children of the node.
subschemaSubentry	A special entry that provides information in the form of operational attributes about the schema elements defined in the server. It identifies the location of the schema for that part of the tree. ldapSyntaxes - set of attribute syntaxes matchingRules - set of matching rules matchingRuleUse - set of matching rule uses attributeTypes - set of attribute types objectClasses - set of object classes nameForms - set of name forms dITContentRules - set of DIT content rules dITStructureRules - set of DIT structure rules

Virtual Attributes	Description
User Defined Virtual Attribute	Generates virtual attributes with user-defined values in entries that match the criteria defined in the plug-in's configuration. User-defined virtual attributes are intended to specify a hard-coded value for entries matching a given set of criteria.
Virtual Static Member	Generates a member attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the `Virtual Static Member` attribute as it is an advanced property and modifying it can lead to undesirable side effects.
Virtual Static Uniquemember	Generates a `uniqueMember` attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the `Virtual Static Uniquemember` attribute as it is an advanced property and modifying it can lead to undesirable side effects.

To View the List of Default Virtual Attributes Using dsconfig Non-Interactive Mode

- Use `dsconfig` to view the virtual attributes.

```
$ bin/dsconfig list-virtual-attributes
```

Viewing Virtual Attribute Properties

Each virtual attribute has basic properties that you can view using the `dsconfig` tool. The complete list of properties is described in the *PingDirectory Server Configuration Reference*. Some basic properties are as follows:

- **Description**. A description of the virtual attribute.
- **Enabled**. Specifies whether the virtual attribute is enabled for use.
- **Base-DN**. Specifies the base DNs for the branches containing entries that are eligible to use this virtual attribute. If no values are given, the server generates virtual attributes anywhere in the server.
- **Group-DN**. Specifies the DNs of the groups whose members can use this virtual attribute. If no values are given, the group membership is not taken into account when generating the virtual attribute. If one or more group DNs are specified, then only members of those groups are allowed to have the virtual attribute.
- **Filter**. Specifies the filters that the server applies to entries to determine if they require virtual attributes. If no values are given, then any entry is eligible to have a virtual attribute value generated.

To View Virtual Attribute Properties

- Use `dsconfig` to view the properties of a virtual attribute.

```
$ bin/dsconfig get-virtual-attribute-prop --name isMemberOf
```

Enabling a Virtual Attribute

You can enable a virtual attribute using the `dsconfig` tool. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Enable a Virtual Attribute using dsconfig Interactive Mode

1. Use `dsconfig` to enable a virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your Directory Server: 1 for LDAP, 2 for SSL, 3 for StartTLS.

   ```
   bin/dsconfig
   ```
2. On the Directory Server main menu, type `o` to change the object menu, and then type the number corresponding to **Standard**.
3. On the Directory Server main menu, type the number corresponding to virtual attributes.
4. On the **Virtual Attribute management** menu, type the number to view and edit an existing virtual attribute.
5. From the list of existing virtual attributes on the system, select the virtual attribute to work with. For this example, type the number corresponding to the `numSubordinates` virtual attribute.
6. On the **numSubordinates Virtual Attribute Properties** menu, type the number to enable the virtual attribute. On the **Enabled Property** menu for the `numSubordinates` virtual attribute, type the number to change the value to TRUE.
7. On the **numSubordinates Virtual Attribute Properties** menu, type `f` to apply the changes.
8. Verify that the virtual attribute is enabled. Note that this example assumes you have configured the group entries.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(ou=People)" numSubordinates
   ```

   ```
   dn: ou=People,dc=example,dc=com
   numSubordinates: 1000
   ```

To Enable a Virtual Attribute Using dsconfig Non-Interactive Mode

- Use `dsconfig` to enable the `numSubordinates` virtual attribute.

  ```
  $ bin/dsconfig set-virtual-attribute-prop \
    --name numSubordinates --set enabled:true
  ```

Creating User-Defined Virtual Attributes

User-defined virtual attributes allow you to specify an explicit value to use for the virtual attribute. There are no restrictions on the length of the value for a user-defined virtual attribute. You must only ensure that the new virtual attribute conforms to your schema, otherwise you will see an error message when you configure it.

You can define your virtual attributes using the `dsconfig` tool on the Standard object menu. Only the value property is specific to the user-defined virtual attribute. All the other properties are common across all kinds of virtual attributes, which include the following:

- **enabled** -- Indicates whether the virtual attribute should be used.
- **attribute-type** -- The attribute type for the virtual attribute that will be generated.
- **base-dn**, **group-dn**, **filter** -- May be used to select which entries are eligible to contain the virtual attribute.
- **client-connection-policy** -- May be used to configure who can see the virtual values.
- **conflict-behavior** -- Used to indicate how the server should behave if there are one or more real values for the same attribute type in the same entry. The server can either return only the real value(s), only the virtual value(s), or merge both real and virtual values.
- **require-explicit-request-by-name** -- Used to indicate whether the server should only generate values for the virtual attribute if it was included in the list of requested attributes.
- **multiple-virtual-attribute-evaluation-order-index**, **multiple-virtual-attribute-merge-behavior** -- Used to control the behavior the server should exhibit if multiple virtual attributes may be used to contribute values to the same attribute.

To Create a User-Defined Virtual Attribute in Interactive Mode

The following example shows how to create a user-defined virtual attribute that assigns an Employee Password Policy to any entry that matches the filter "`(employeeType=employee)`".

1. Run `dsconfig` to configure the user-defined virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your Directory Server: 1 for LDAP, 2 for SSL, 3 for StartTLS.
2. On the Directory Server main menu, type `o` to change the object menu, and then type the number to select Standard.
3. On the Directory Server main menu, type the number corresponding to virtual attributes.
4. On the **Virtual Attribute management** menu, type the number to create a new virtual attribute.
5. Next, you can use an existing virtual attribute as a template for your new attribute, or your can create a new attribute from scratch. In this example, type `n` to create a new Virtual Attribute from scratch.
6. On the **Virtual Attribute Type** menu, enter a number corresponding to the type of virtual attribute that you want to create. In this example, type the number corresponding to User Defined Virtual Attribute.
7. Next, enter a name for the new virtual attribute. In this example, enter "`Employee Password Policy Assignment`."
8. On the **Enabled Property** menu, enter the number to set the property to true (enable).
9. On the **Attribute-Type Property** menu, type the `attribute-type` property for the new virtual attribute. You can enter the OID number or attribute name. The `attribute-type` property must conform to your schema. For this example, type "`ds-pwp-password-policy-dn`".
10. Enter the value for the virtual attribute, and then press Enter or Return to continue. In this example, enter `cn=Employee Password Policy,cn=Password Policies,cn=config`, and then type Enter or Return to continue.
11. On the **User Defined Virtual Attributes** menu, enter a description for the virtual attribute. Though optional, this step is useful if you plan to create a lot of virtual attributes. Enter the option to change the value, and then type a description of the virtual attribute. In this example, enter: `Virtual attribute that assigns the Employee Password Policy to all entries that match (employeeType=employee).`
12. On the **User Defined Virtual Attribute** menu, type the number corresponding to the filter.
13. On the **Filter Property** menu, enter the option to add one or more filter properties, type the filter, and then press **Enter** to continue. In this example, type `(employeeType=employee)`. Press the number to use the filter value entered.
14. On the **User Defined Virtual Attribute** menu, type `f` to finish creating the virtual attribute.
15. Verify that the attribute was created successfully. Add the `employeeType=employee` attribute to an entry (e.g., `uid=user.0`) using `ldapmodify`. Add the `employeeType=contractor` attribute to another entry (e.g., `uid=user.1`).
16. Use `ldapsearch` to search for the user with the `employeeType=employee` attribute (e.g., `uid=user.0`). You will notice the `ds-pwp-password-policy-dn` attribute has the assigned password policy as its value.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
  ds-password-policy-dn
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Employee Password Policy,cn=Password
 Policies,cn=config
```

17. Run `ldapsearch` again using the filter "`(uid=user.1)`", the `ds-pwp-password-policy-dn` attribute will not be present in the entry, because the entry has the attribute, `employeeType=contractor`.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.1)" \
  ds-password-policy-dn
```

```
dn: uid=user.1,ou=People,dc=example,dc=com
```

To Create a User-Defined Virtual Attribute Using dsconfig in Non-Interactive Mode

- You can also use `dsconfig` in non-interactive command-line mode to create a virtual attribute. The following command sets up the Employee Password Policy Assignment virtual attribute introduced in the previous section:

```
$ bin/dsconfig create-virtual-attribute \
  --name "Employee Password Policy Assignment" \
  --type user-defined \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "filter:(employeeType=employee)" \
  --set "value:cn=Employee Password Policy,cn=Password Policies,cn=config"
```

Creating Mirror Virtual Attributes

The PingDirectory Server provides a feature to mirror the value of another attribute in the same entry or mirror the value of the same or a different attribute in an entry referenced by the original entry. For example, given a DIT where users have a manager attributed with a value of the DN of the employee as follows:

```
dn: uid=apeters,ou=people,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
uid: apeters
... (more attributes) ...
```

You can set up a mirror virtual attribute, so that the returned value for the `managerName` virtual attribute can be the `cn` value of the entry referenced by the `manager` attribute as follows:

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=apeters)" \
dn: uid=apeters,ou=people,dc=example,dc=com

objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
managerName: John Doe
uid: apeters
... (more attributes not shown) ...
```

To Create a Mirror Virtual Attribute in Non-Interactive Mode

You can also use `dsconfig` in non-interactive command-line mode to create a mirror virtual attribute. The following example sets up the `managerName` virtual attribute introduced in the previous section:

1. Update the schema to define the `managerName` attribute. In a text editor, create a file with the following schema definition for the attribute and save it as `98-myschema.ldif`, for example, in the `<server-root>/config/schem`a folder. You can optionally add the attribute to an object class.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubent
ry
objectClass: subschema attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.4 NAME
 ‘managerName’
  EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX
 1.3.6.1.4.1.1466.115.121.1.44{256}
  X-ORIGIN ‘Directory Server Example’ )
```

2. Restart the Directory Server.

   ```
   $ bin/stop-server --restart
   ```

3. Use `dsconfig` to create the virtual attribute.

   ```
   $ bin/dsconfig create-virtual-attribute \
     --name "managerName" \
     --type mirror \
     --set "description:managerName from manager cn" \
     --set enabled:true \
     --set attribute-type:managerName \
     --set source-attribute:cn \
     --set source-entry-dn-attribute:manager
   ```

4. Verify the mirror virtual attribute by searching for an entry.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=apeters)"
   ```

   ```
   dn: uid=apeters,ou=People,dc=example,dc=com
   ... (attributes) ...
   manager: uid=jdoe,ou=People,dc=example,dc=com
   managerName: John Doe
   ```

Editing a Virtual Attribute

You can edit virtual attributes using the `dsconfig` tool. You must ensure that the virtual attribute conforms to your plug-in schema, otherwise you will see an error message when you edit the virtual attribute. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Edit a Virtual Attribute Using dsconfig in Non-Interactive Mode

- Use `dsconfig` to change a property’s value.

  ```
  $ bin/dsconfig set-virtual-attribute-prop --name dept-number \
    --set "value:111"
  ```

Deleting a Virtual Attribute

You can delete virtual attributes using the `dsconfig` tool. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Delete a Virtual Attribute

- Use `dsconfig` to delete an existing virtual attribute.

  ```
  $ bin/dsconfig delete-virtual-attribute --name dept-number
  ```

Chapter

13

Working with Groups

Topics:

LDAP groups are special types of entries that represent collections of users. Groups are often used by external clients, for example, to control who has access to a particular application or features. They may also be used internally by the server to control its behavior. For example, groups can be used by the access control, criteria, or virtual attribute subsystems.

The specific ways in which clients create and interact with a particular group depends on the type of group being used. In general, there are three primary ways in which clients attempt to use groups:

To determine whether a specified user is a member of a particular group.
To determine the set of groups in which a specified user is a member.
To determine the set of all users that are members of a particular group.

This chapter provides an overview of Directory Server groups concepts and provides procedures on setting up and querying groups in the Directory Server.

Overview of Groups

The Directory Server provides the following types of groups:

- **Static Groups**. A static group is an entry that contains an explicit list of member or uniquemember attributes, depending on its particular structural object class. Static groups are ideal for relatively small, infrequently changing elements. Once the membership list grows, static groups become more difficult to manage as any change in a member base DN must also be changed in the group. Static groups use one of three structural object classes: `groupOfNames`, `groupOfUniqueNames`, and `groupOfEntries`.

  The Directory Server also supports nested groups, in which a parent group entry contains child attributes whose DNs reference another group. Nested groups are a flexible means to organize entries that provide inherited group membership and privileges. To maintain good performance throughput, a group cache is enabled by default. The cache supports static group nesting that includes other static, virtual static, and dynamic groups.
- **Dynamic Groups**. A dynamic group has its membership list determined by search criteria using a LDAP URL. Dynamic groups solve the scalability issues encountered for static groups as searches are efficient, constant-time operations. However, if searches range over a very large set of data, performance could be affected.
- **Virtual Static Groups**. A virtual static group is a combination of both static and dynamic groups, in which each member in a group is a virtual attribute that is dynamically generated when invoked. Virtual static groups solve the scalability issues for clients that can only support static groups and are best used when the application targets a search operation for a specific member. Virtual static groups are not good for applications that need to retrieve the entire membership list as the process for constructing the entire membership list can be expensive.

About the isMemberOf and isDirectMemberOf Virtual Attribute

The existence of both static, nested, dynamic, and virtual static groups can make it unnecessarily complex to work with groups in the server, particularly because the ways you interact with them are so different. And the fact that static groups can use three different structural object classes (not counting the auxiliary class for virtual static groups) does not make things any easier.

To make group operations simpler, the PingDirectory Server provides the ability to generate either an `isMemberOf` and `isDirectMemberOf` virtual attributes in user entries. These attributes dramatically simplify the process for making group-related determinations in a manner that is consistent across all types of groups.

The value of the `isMemberOf` virtual attribute is a list of DNs of all groups (including static, nested, dynamic, and virtual static groups) in which the associated user is a member. The value of the `isDirectMemberOf` virtual attribute is a subset of the values of `isMemberOf`, which represents the groups for which the entry is an explicit or direct member. Both are enabled by default.

Because the `isMemberOf` and `isDirectMemberOf` are operational attributes, only users who specifically have been granted the privilege can see it. The default set of access control rules do not allow any level of access to user data. The only access that is granted is what is included in user-defined access control rules, which is generally given to a `uid=admin` administrator account. It is always a best practice to restrict access to operational and non-operational attributes to the minimal set of users that need to see them. The root bind DN, `cn=Directory Manager`, has the privilege to view operational attributes by default.

To determine whether a user is a member of a specified group using the `isMemberOf` virtual attribute, simply perform a base-level search against the user's entry with an equality filter targeting the `isMemberOf` attribute with a value that is the DN of the target group. The following table illustrates this simple base-level search:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)
Requested Attributes	1.1

If this search returns an entry, then the user is a member of the specified group. If no entry is returned, then the user is not a member of the given group.

To determine the set of all groups in which a user is a member, simply retrieve the user's entry with a base-level search and include the `isMemberOf` attribute:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(objectclass=*)
Requested Attributes	isMemberOf

To determine the set of all members for a specified group, issue a subtree search with an equality filter targeting the `isMemberOf` attribute with a value that is the DN of the target group and requesting the attributes you wish to have for member entries:

Base DN	ou=People,dc=example,dc=com
Scope	sub
Filter	(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)
Requested Attributes	cn, mail

The `isDirectMemberOf` virtual attribute can be used in the examples above in place of `isMemberOf` if you only need to find groups that users are an actual member of. You must use `isMemberOf` for nested group membership.

Note that if this filter targets a dynamic group using an unindexed search, then this may be an expensive operation. However, it will not be any more expensive than retrieving the target group and then issuing a search based on information contained in the member URL.

For static groups, this approach has the added benefit of using a single search to retrieve information from all user entries, whereas it would otherwise be required to retrieve the static group and then perform a separate search for each member's entry.

Using Static Groups

A static group contains an explicit membership list where each member is represented as a DN-valued attribute. There are three types of static groups supported for use in the Directory Server:

- **groupOfNames**. A static group that is defined with the `groupOfNames` structural object class and uses the `member` attribute to hold the DNs of its members. RFC 4519 requires that the `member` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `member` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
cn: Test Group
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```

- **groupOfUniqueNames**. A static group that is defined with the `groupOfUniqueNames` structural object class and uses the `uniquemember` attribute to hold the DNs of its members. RFC 4519 requires that the `uniquemember` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `uniquemember` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfUniqueNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: Test Group
uniquemember: uid=user.1,ou=People,dc=example,dc=com
uniquemember: uid=user.2,ou=People,dc=example,dc=com
uniquemember: uid=user.3,ou=People,dc=example,dc=com
```

- **groupOfEntries**. A static group that is defined with the `groupOfEntries` object class and uses the `member` attribute to hold the DNs of its members. This group specifies that the `member` attribute is optional to ensure that the last member can be removed from the group. Although the draft proposal (draft-findlay-ldap-groupofentries-00.txt) has expired, the Directory Server supports this implementation. The following entry depicts a group defined with the `groupOfEntries` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfEntries
cn: Test Group
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```

Creating Static Groups

You can configure a static group by adding it using an LDIF file. Static groups contain a membership list of explicit DNs specified by the `uniquemember` attribute.

To Create a Static Group

1. Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniquenames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your server, then you can add it in the same file. When done, save the file as `static-group.ldif`. The following example LDIF file creates two groups, `cn=Development` and `cn=QA`.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Development,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Development
ou: groups
uniquemember: uid=user.14,ou=People,dc=example,dc=com
uniquemember: uid=user.91,ou=People,dc=example,dc=com
uniquemember: uid=user.180,ou=People,dc=example,dc=com

dn: cn=QA,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: QA
ou: groups
uniquemember: uid=user.0,ou=People,dc=example,dc=com
uniquemember: uid=user.1,ou=People,dc=example,dc=com
uniquemember: uid=user.2,ou=People,dc=example,dc=com
```

2. Use `ldapmodify` to add the group entries to the server.

```
$ bin/ldapmodify --defaultAdd --filename static-group.ldif
```

3. Verify the configuration by using the virtual attribute `isDirectMemberOf` that checks membership for a non-nested group. By default, the virtual attribute is disabled by default, but you can enable it using `dsconfig`.

```
$ bin/dsconfig set-virtual-attribute-prop --name isDirectMemberOf --set
 enabled:true
```

4. Use `ldapsearch` to specifically search the `isDirectMemberOf` virtual attribute to determine if `uid=user.14` is a member of the `cn=Development` group. In this example, assume that administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isDirectMemberOf
```

```
dn: uid=user.14,ou=People,dc=example,dc=com
isDirectMemberOf: cn=Development,ou=groups,dc=example,dc=com
```

5. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an `aci` attribute in an LDIF file, and save the file as `dev-group-aci.ldif`. Add the file using the `ldapmodify` tool. You can create a similar ACI for the QA group, which is not shown in this example.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr != "cn || sn || uid")
  (targetfilter ="(ou=Development)")
  (version 3.0; acl "Dev Group Permissions";
    allow (write) (groupdn = "ldap:///
cn=Development,ou=groups,dc=example,dc=com");)
```

6. Add the file using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename dev-group-aci.ldif
```

To Add a New Member to a Static Group

- To add a new member to the group, add a new value for the `uniquemember` attribute that specifies the DN of the user to be added. The following example adds a new `uniquemember`, `user.4`:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
add: uniquemember
uniquemember: uid=user.4,ou=People,dc=example,dc=com
```

To Remove a Member from a Static Group

- To remove a member from a static group, remove that user's DN from the `uniquemember` attribute. The following example removes the DN of `user.1`:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
delete: uniquemember
uniquemember: uid=user.1,ou=People,dc=example,dc=com
```

Searching Static Groups

The following sections describe how to compose searches to determine if a user is a member of a static group, to determine all the static groups in which a user is a member, and to determine all the members of a static group.

To Determine if a User is a Static Group Member

To determine whether a user is a member of a specified group, perform a base-level search to retrieve the group entry with an equality filter looking for the membership attribute of a value equal to the DN of the specified user.

For best performance, you will want to include a specific attribute list (just "cn", or "1.1" request that no attributes be returned) so that the entire member list is not returned. For example, to determine whether the user "uid=john.doe,ou=People,dc=example,dc=com" is a member of the groupOfNames static group "cn=Test Group,ou=Groups,dc=example,dc=com", issue a search with the following criteria:

Table 20: Search Criteria for a Single User's Membership in a Static Group

Base DN	cn=Test Group,ou=Groups,dc=example,dc=com
Scope	base
Filter	(member=uid=john.doe,ou=People,dc=example,dc=com)
Requested Attributes	1.1

If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group. If you do not know the membership attribute for the specified group (it could be either a member or uniqueMember attribute), then you may want to revise the filter so that it allows either one as follows:

```
(|(member=uid=john.doe,ou=People,dc=example,dc=com)
(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com))
```

- Run a base-level search to retrieve the group entry with an equality filter looking for the membership attribute.

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com"
  --searchScope base "(member=uid=john.doe,ou=People,dc=example,dc=com)"
 "1.1"
```

To Determine the Static Groups to Which a User Belongs

To determine the set of all static groups in which a user is specified as a member, perform a subtree search based at the top of the DIT. The search filter must be configured to match any type of static group in which the specified user is a member.

For example, the following criteria may be used to determine the set of all static groups in which the user, uid=john.doc,ou=People,dc=example,dc=com, is a member:

Table 21: Search Criteria for Determining All the Static Groups for a User

Base DN	dc=example,dc=com
Scope	sub
Filter	(|(&(objectClass=groupOfNames)(member=uid=john.doe,ou=People,dc=example,dc=com))(&(objectClass=groupOfUniqueNames)(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com)) (&(objectClass=groupOfEntries)(member=uid=john.doe,ou=People,dc=example,dc=com)))
Requested Attributes	1.1

Every entry returned from the search represents a static group in which the specified user is a member.

- Run a sub-level search to retrieve the static groups to which a user belongs.

```
$ bin/ldapsearch --baseDN "dc=example,dc=com" --searchScope sub \
  "(|(&(objectClass=groupOfNames)
  (member=uid=john.doe,ou=People,dc=example,dc=com)) \
  (&(objectClass=groupOfUniqueNames)\
  (uniqueMember=uid=john.doe,ou=People,dc=example,dc=com)) \
  (&(objectClass=groupOfEntries) \
  (member=uid=john.doe,ou=People,dc=example,dc=com)))" "1.1"
```

Note: A base level search of the user's entry for `isMemberOf` or `isDirectMemberOf` virtual attributes will give the same results. You can also use the virtual attributes with virtual static groups.

To Determine the Members of a Static Group

To determine all of the members for a static group, simply retrieve the group entry including the membership attribute. The returned entry will include the DNs of all users that are members of that group. For example, the following criteria may be used to retrieve the list of all members for the group `cn=Test Group,ou=Groups,dc=example,dc=com`:

Table 22: Search Criteria for All of the Static Group's Members

Base DN	cn=Test Group,ou=Groups,dc=example,dc=com
Scope	base
Filter	(objectClass=*)
Requested Attributes	member uniqueMember

If you want to retrieve additional information about the members, such as attributes from member entries, you must issue a separate search for each member to retrieve the user entry and the desired attributes.

- Run a base-level search to retrieve all of the members in a static group.

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com" \
  --searchScope base "(objectclass=*)" uniqueMember
```

Note: If you want to retrieve attributes from member entries, it is more efficient to search all users whose `isMemberOf` attribute contains the group DN, returning the attributes desired.

Using Dynamic Groups

Dynamic groups contain a set of criteria used to identify members rather than maintaining an explicit list of group members. If a new user entry is created or if an existing entry is modified so that it matches the membership criteria, then the user will be considered a member of the dynamic group. Similarly, if a member's entry is deleted or if it is modified so that it no longer matches the group criteria, then the user will no longer be considered a member of the dynamic group.

In the Directory Server, dynamic groups include the `groupOfURLs` structural object class and use the `memberurl` attribute to provide an LDAP URL that defines the membership criteria. The base, scope, and filter of the LDAP URL will be used in the process of making the determination, and any other elements present in the URL will be ignored. For example, the following entry defines a dynamic group in which all users below `dc=example,dc=com` with an `employeeType` value of contractor will be considered members of the group:

```
dn: cn=Sales Group,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Sales Group
memberURL: ldap:///dc=example,dc=com??sub?(employeeType=contractor)
```

Assuming that less than 80,000 entries have the `employeeType` of contractor, you need to create the following index definition to evaluate the dynamic group:

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name employeeType --set index-entry-limit:80000 \
  --set index-type:equality
```

Creating Dynamic Groups

You can configure a dynamic group in the same manner as static groups using an LDIF file. Dynamic groups contain a membership list of attributes determined by search filter using an LDAP URL. You must use the `groupOfURLs` object class and the `memberURL` attribute.

To Create a Dynamic Group

1. Assume that `uid=user.15` is not part of any group. Use `ldapsearch` to verify that `uid=user.15` is not part of any group. In a later step, we will add the user to the dynamic group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub
 "(uid=user.15)" ou
```

```
dn: uid=user.15,ou=People,dc=example,dc=com
```

2. Assume for this example that `uid=user.0` has an `ou=Engineering` attribute indicating that he or she is a member of the Engineering department.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.0)"
 ou isMemberOf
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ou: Engineering
```

3. Open a text editor, and then create a dynamic group entry in LDIF. The LDIF defines the dynamic group to include all users who have the `ou=Engineering` attribute. When done, save the file as `add-dynamic-group.ldif`.

```
dn: cn=eng-staff,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfURLs
ou: groups
cn: eng-staff
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(ou=Engineering)
```

4. Use `ldapmodify` to add the group entry to the server.

```
$ bin/ldapmodify --defaultAdd --filename add-dynamic-group.ldif
```

5. Use `ldapsearch` to specifically search the `isMemberOf` virtual attribute to determine if `uid=user.0` is a member of the `cn=Engineering` group or any other group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" isMemberOf
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

6. If your data is relatively small (under 1 million entries), you can search for all users in the group that meet the search criteria (`ou=Engineering`). For very large databases, it is not practical to run a database-wide search for all users as there can be a performance hit on the Directory Server. The following command returns the DNs of entries that are part of the `cn=eng-staff` dynamic group and sorts them in ascending order by the `sn` attribute.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --sortOrder sn \
  "(isMemberOf=cn=eng-staff,ou=groups,dc=example,dc=com)" dn
```

7. Add `uid=user.15` to the `eng-staff` group by adding an `ou=Engineering` attribute to the entry. This step highlights an advantage of dynamic groups: you can make a change in an entry without explicitly adding the DN to the group as you would with static groups. The entry will be automatically added to the `eng-staff` dynamic group.

```
$ bin/ldapmodify
dn: uid=user.15,ou=People,dc=example,dc=com
changetype: modify
```

```
add: ou
ou: Engineering
```

8. Use `ldapsearch` to check if the user is part of the `cn=eng-staff` dynamic group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub
 "(uid=user.15)" isMemberOf
```

```
dn: uid=user.15,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

Searching Dynamic Groups

The following sections describe how to compose searches to determine if a user is a member of a dynamic group, to determine all the dynamic groups in which a user is a member, and to determine all the members of a dynamic group.

To Determine if a User is a Dynamic Group Member

To determine whether a user is a member of a specific dynamic group, you must verify that the user's entry is both within the scope of the member URL and that it matches the filter contained in that URL. You can verify that a user's entry is within the scope of the URL using simple client-side only processing. Evaluating the filter against the entry on the client side can be more complicated. While possible, particularly in clients that are able to perform schema-aware evaluation, a simple alternative is to perform a base-level search to retrieve the user's entry with the filter contained in the member URL.

For example, to determine whether the user `uid=john.doe,ou=People,dc=example,dc=com` is a member of the dynamic group with the above member URL, issue a search with the following criteria:

Table 23: Search Criteria for a Single User's Membership in a Dynamic Group

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(ou=Engineering)
Requested Attributes	1.1

Note that the search requires the user DN to be under the search base defined in the `memberurl` attribute for the user to be a member. If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group.

To Determine the Dynamic Groups to Which a User Belongs

To determine the set of all dynamic groups in which a user is a member, first perform a search to find all dynamic group entries defined in the server. You can do this using a subtree search with a filter of `"(objectClass=groupOfURLs)"`.

You should retrieve the `memberURL` attribute so that you can use the logic described in the previous section to determine whether the specified user is a member of each of those groups. For example, to find the set of all dynamic groups defined in the `dc=example,dc=com` tree, issue a search with the following criteria:

Table 24: Search Criteria for Determining All of the Dynamic Groups for a User

Base DN	dc=example,dc=com
Scope	sub
Filter	(objectClass=groupOfURLs)
Requested Attributes	memberURL

Each entry returned will be a dynamic group definition. You can use the base, scope, and filter of its `memberURL` attribute to determine whether the user is a member of that dynamic group.

To Determine the Members of a Dynamic Group

To determine all members of a dynamic group, issue a search using the base, scope, and filter of the member URL. The set of requested attributes should reflect the attributes desired from the member user entries, or "1.1" if no attributes are needed.

For example, to retrieve the `cn` and `mail` attributes from the group described above, use the following search:

Table 25: Search Criteria for Determining the Members of a Dynamic Group

Base DN	dc=example,dc=com
Scope	sub
Filter	(employeeType=contractor)
Requested Attributes	cn, mail

⚠ **CAUTION:** Note that this search may be expensive if the associated filter is not indexed or if the group contains a large number of members.

Using Dynamic Groups for Internal Operations

You can use dynamic groups for internal operations, such as ACI or component evaluation. The Directory Server performs the `memberurl` parsing and internal LDAP search; however, the internal search operation may not be performed with access control rules applied to it.

For example, the following dynamic group represents an organization’s employees within the same department:

```
dn: cn=department 202,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: department 202
owner: uid=user.1,ou=people,dc=example,dc=com
owner: uid=user.2,ou=people,dc=example,dc=com
memberURL: ldap:///ou=People,dc=example,dc=com??sub?
  (&(employeeType=employee)(departmentNumber=202))
description: Group of employees in department 202
```

The above group could be referenced from within the ACI at the `dc=example,dc=com` entry. For example:

```
dn:dc=example,dc=com
aci: (targetattr="employeeType")
  (version 3.0; acl "Grant write access to employeeType" ;
    allow (all) groupdn="ldap:///cn=department
 202,ou=groups,dc=example,dc=com";)
```

Any user matching the filter can bind to the server with their entry and modify the `employeeType` attribute within any entry under `dc=example,dc=com`.

Using Virtual Static Groups

Static groups can be easier to interact with than dynamic groups, but large static groups can be expensive to manage and require a large amount of memory to hold in the internal group cache. The Directory Server provides a third type of group that makes it possible to get the efficiency and ease of management of a dynamic group while allowing clients to interact with it as a static group. A *virtual static group* is a type of group that references another group and provides access to the members of that group as if it was a static group.

To create a virtual static group, create an entry that has a structural object class of either groupOfNames or groupOfUniqueNames and an auxiliary class of ds-virtual-static-group. It should also include a ds-target-group-dn attribute, whose value is the group from which the virtual static group should obtain its members. For example, the following will create a virtual static group that exposes the members of the cn=Sales Group,ou=Groups,dc=example,dc=com dynamic group as if it were a static group:

```
dn: cn=Virtual Static Sales Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
objectClass: ds-virtual-static-group
cn: Virtual Static Sales Group
ds-target-group-dn: cn=Sales Group,ou=Groups,dc=example,dc=com
```

Note that you must also enable a virtual attribute that allows the member attribute to be generated based on membership for the target group. A configuration object for this virtual attribute does exist in the server configuration, but is disabled by default. To enable it, issue the following change:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set enabled:true
```

If you want to use virtual static groups with the groupOfUniqueNames object class, then you will also need to enable the Virtual Static uniqueMember virtual attribute in the same way.

Creating Virtual Static Groups

If your application only supports static groups but has scalability issues, then using a virtual static group could be a possible solution. A virtual static group uses a virtual attribute that is dynamically generated when called after which the operations that determine group membership are passed to another group, such as a dynamic group. You must use the ds-virtual-static-group object class and the ds-target-group-dn virtual attribute.

Virtual static groups are best used when determining if a single user is a member of a group. It is not a good solution if an application accesses the full list of group members due to the performance expense at constructing the list. If you have a small database and an application that requires that the full membership list be returned, you must also enable the allow-retrieving-membership property for the Virtual Static uniqueMember virtual attribute using the dsconfig tool.

To Create a Virtual Static Group

1. Open a text editor, and then create a group entry in LDIF. The entry contains the groupOfUniqueNames object class, but in place of the uniquemember attribute is the ds-target-group-dn virtual attribute, which is part of the ds-virtual-static-group auxiliary object class. When done, save the file as add-virtual-static-group.ldif.

```
dn: cn=virtualstatic,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
objectclass: ds-virtual-static-group
ou: groups
cn: virtual static
ds-target-group-dn: cn=eng-staff,ou=groups,dc=example,dc=com
```

2. Use ldapmodify to add the virtual static group entry to the server.

```
$ bin/ldapmodify -h server1.example.com -p 389 -D
 "uid=admin,dc=example,dc=com" \
  -w password -a -f add-virtual-static-group.ldif
```

3. Use dsconfig to enable the Virtual Static uniqueMember attribute, which is disabled by default.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
 uniqueMember" \
  --set enabled:true
```

4. In the previous section, we set up `uid=user.0` to be part of the `cn=eng-staff` dynamic group. Use `ldapsearch` with the `isMemberOf` virtual attribute to determine if `uid=user.0` is part of the virtual static group.

```
$ bin/ldapsearch -h server1.example.com -p 389 -D "cn=Directory Manager" \
  -w secret -b dc=example,dc=com" "(uid=user.0)" isMemberOf
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
isMemberOf: cn=virtualstatic,ou=groups,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

5. Use `ldapsearch` to determine if `uid=user.0` is a member of the virtual static group. You should see the returned `cn=virtualstatic` entry if successful.

```
$ ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w password \
  -b "cn=virtualStatic,ou=Groups,dc=example,dc=com" \
  "(&(objectclass=groupOfUniqueNames) \
  (uniquemember=uid=user.0,ou=People,dc=example,dc=com))"
```

6. Next, try searching for a user that is not part of the `cn=eng-staff` dynamic group (e.g., `uid=user.20`), nothing will be returned.

```
$ ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w password \
  -b "cn=virtualStatic,ou=Groups,dc=example,dc=com" \
  "(&(objectclass=groupOfUniqueNames) \
  (uniquemember=uid=user.20,ou=People,dc=example,dc=com))"
```

Searching Virtual Static Groups

Because virtual static groups behave like static groups, the process for determining whether a user is a member of a virtual static group is identical to that of a member in a static group. Similarly, the process for determining all virtual static groups in which a user is a member is basically the same as the process as that of real static groups in which a user is a member. In fact, the query provided in the static groups discussion returns virtual static groups in addition to real static groups, because the structural object class of a virtual static group is the same as the structural object class for a static group.

You can also retrieve a list of all members of a virtual static group in the same way as a real static group: simply retrieve the `member` or `uniqueMember` attribute of the desired group. However, because virtual static groups are backed by dynamic groups and the process for retrieving member information for dynamic groups can be expensive, virtual static groups do not allow retrieving the full set of members by default. The virtual attribute used to expose membership can be updated to allow this with a configuration change such as the following:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set allow-retrieving-membership:true
```

Because this can be an expensive operation, we recommend that the option to allow retrieving virtual static group membership be left disabled unless it is required.

Creating Nested Groups

The PingDirectory Server supports nested groups, where the DN of an entry that defines a group is included as a member in the parent entry. For example, the following example shows a nested static group (e.g., `cn=Engineering Group`) that has `uniquemember` attributes consisting of other groups, such as `cn=Developers Group` and the `cn=QA Group` respectively.

```
dn: cn=Engineering Group,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers,ou=Groups,dc=example,dc=com
```

```
uniquemember: cn=QA,ou=Groups,dc=example,dc=com
```

Nested group support is enabled by default on the Directory Server. To support nested groups without the performance hit, the Directory Server uses a group cache, which is also enabled by default. The cache supports static group nesting that includes other static, virtual static, and dynamic groups. The Directory Server provides a new monitoring entry for the group cache, `cn=Group Cache,cn=Monitor`.

In practice, nested groups are not commonly used for several reasons. LDAP specifications do not directly address the concept of nested groups, and some servers do not provide any level of support for them. Supporting nested groups in LDAP clients is not trivial, and many directory server-enabled applications that can interact with groups do not provide any support for nesting. If nesting support is not needed in your environment, or if nesting support is only required for clients but is not needed for server-side evaluation (such as for groups used in access control rules, criteria, virtual attributes, or other ways that the server may need to make a membership determination), then this support should be disabled.

To Create Nested Static Groups

1. The following example shows how to set up a nested static group, which is a static group that contains `uniquemember` attributes whose values contain other groups (static, virtual static, or dynamic). Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniquenames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your server, then you can add it in the same file. When done, save the file as `nested-group.ldif`. Assume that the static groups, `cn=Developers Group` and `cn=QA Group`, have been configured.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Engineering Group,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers,ou=groups,dc=example,dc=com
uniquemember: cn=QA,ou=groups,dc=example,dc=com
```

2. Use `ldapmodify` to add the group entry.

```
$ bin/ldapmodify --defaultAdd --filename nested-static-group.ldif
```

3. Verify the configuration by using the `isMemberOf` virtual attribute that checks the group membership for an entry. By default, the virtual attribute is enabled. Use `ldapsearch` to specifically search the `isMemberOf` virtual attribute to determine if `uid=user.14` is a member of the `cn=Development` group. In this example, assume that the administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isMemberOf

dn: uid=user.14,ou=People,dc=example,dc=com
isMemberOf: cn=Development,ou=groups,dc=example,dc=com
```

4. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an ACI in LDIF, and save the file as `eng-group-aci.ldif`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr != "cn || sn || uid")
  (targetfilter ="(ou=Engineering Group)")
  (version 3.0; acl "Engineering Group Permissions";
    allow (write) (groupdn = "ldap:///cn=Engineering
 Group,ou=groups,dc=example,dc=com");)
```

5. Add the file using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename eng-group-aci.ldif
```

Note: When nesting dynamic groups, you cannot include other groups as members of a dynamic group. You can only support "nesting" by including the members of another group with a filter in the member URL. For example, if you have two groups `cn=dynamic1` and `cn=dynamic2`, you can nest one group in another by specifying it in the member URL as follows:

```
cn=dynamic1,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
memberURL: ldap:///dc=example,dc=com??sub?
(isMemberOf=cn=dynamic2,ou=groups,dc=example,dc=com)
```

The members included from the other group using this method are not considered "nested" members and will be returned even when using `isDirectMemberOf` when retrieving the members.

Maintaining Referential Integrity with Static Groups

The Directory Server can automatically update references to an entry whenever that entry is removed or renamed in a process called *referential integrity*. For example, if a user entry is deleted, then referential integrity plugin will remove that user from any static groups in which the user was a member (this is not necessary for dynamic groups, since no explicit membership is maintained). Similarly, if a modify DN operation is performed to move or rename a user entry, then referential integrity updates static groups in which that user is a member with the new user DN.

Referential integrity support is disabled by default, but may be enabled using the `dsconfig` tool as follows:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

Other configuration attributes of note for this plugin include:

- **attribute-type**. This attribute specifies the names or OIDs of the attribute types for which referential integrity will be maintained. By default, referential integrity is maintained for the `member` and `uniqueMember` attributes. Any attribute types specified must have a syntax of either distinguished name (OID "1.3.6.1.4.1.1466.115.121.1.12") or name and optional UID (OID "1.3.6.1.4.1.1466.115.121.1.34"). The specified attribute types must also be indexed for equality in all backends for which referential integrity is to be maintained.
- **base-dn**. This attribute specifies the subtrees for which referential integrity will be maintained. If one or more values are provided, then referential integrity processing will only be performed for entries which exist within those portions of the DIT. If no values are provided (which is the default behavior), then entries within all public naming contexts will be included.
- **log-file**. This attribute specifies the path to a log file that may be used to hold information about the DNs of deleted or renamed entries. If the plugin is configured with a nonzero update interval, this log file helps ensure that appropriate referential integrity processing occurs even if the server is restarted.
- **update-interval**. This attribute specifies the maximum length of time that a background thread may sleep between checks of the referential integrity log file to determine whether any referential integrity processing is required. By default, this attribute has a value of "0 seconds", which indicates that all referential integrity processing is to be performed synchronously before a response is returned to the client. A duration greater than 0 seconds indicates that referential integrity processing will be performed in the background and will not delay the response to the client.

In the default configuration, where referential integrity processing is performed synchronously, the throughput and response time of delete and modify DN operations may be adversely impacted because the necessary cleanup work must be completed before the response to the original operation can be returned. Changing the configuration to use a non-zero update interval alleviates this performance impact because referential integrity processing uses a separate background thread and does not significantly delay the response to delete or modify DN operations.

However, performing referential integrity processing in a background thread may introduce a race condition that may adversely impact clients that delete a user and then immediately attempt to re-add it and establish new group memberships. If referential integrity processing has not yet been completed for the delete, then newly-established group memberships may be removed along with those that already existed for the previous user. Similarly, if the newly-created user is to be a member of one or more of the same groups as the previous user, then attempts by the client to re-establish those memberships may fail if referential integrity processing has not yet removed the previous membership. For this reason, we recommend that the default synchronous behavior be maintained unless the performance impact associated with it is unacceptable and clients are not expected to operate in a manner that may be adversely impacted by delayed referential integrity processing.

Note: The internal operations of the referential integrity plug-in are not replicated. So, in a replicated topology, you must enable the referential integrity plug-in consistently on all servers in the topology to ensure that changes made by the referential integrity plug-in are passed along to a replication server.

For more information about administering the referential integrity plug-in, see Chapter 6, “Configuring the Directory Server” in the *Ping Identity Directory Server Administration Guide*.

Monitoring the Group Membership Cache

The Directory Server logs information at startup about the memory consumed by the group membership cache. This hard-coded cache contains information about all of the group memberships for internal processing, such as ACIs. The group membership cache is enabled by default.

The information about this cache is logged to the standard output log (`server.out`) and the standard error log. When using groups, you can use the log information to tune the server for best performance. For example, at startup the server logs a message like the following to the `server.out` log:

```
[16/Aug/2011:17:14:39.462 -0500] category=JEB severity=NOTICE msgID=1887895587
msg="The database cache now holds 3419MB of data and is 32 percent full"
```

The error log will contain something like the following:

```
[16/Aug/2011:18:40:39.555 -0500] category=EXTENSIONS severity=NOTICE
 msgID=1880555575
msg="'Group cache (174789 static group(s) with 7480151 total memberships and
 1000002
unique members, 0 virtual static group(s), 1 dynamic group(s))' currently
 consumes
149433592 bytes and can grow to a maximum of 149433592 bytes"
```

Using the Entry Cache to Improve the Performance of Large Static Groups

The PingDirectory Server provides an entry cache implementation, which allows for fine-grained control over the kinds of entries that may be held in the cache. You can define filters to specify the entries included in or excluded from the cache, and you can restrict the cache so that it holds only entries with at least a specified number of values for a given set of attributes.

Under most circumstances, we recommend that the Directory Server be used *without* an entry cache. The Directory Server is designed to efficiently retrieve and decode entries from the database in most cases. The database cache is much more space-efficient than the entry cache, and heavy churn in the entry cache can adversely impact garbage collection behavior.

However, if the Directory Server contains very large static groups, such as those containing thousands or millions of members, and clients need to frequently retrieve or otherwise interact with these groups, then you may want to enable an entry cache that holds only large static groups.

In servers containing large static groups, you can define an entry cache to hold only those large static groups. This entry cache should have an include filter that matches only group entries (for example, "(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfEntries))"). The filter contains a minimum value count so that only groups with a large number of members (such as those with at least 100 member or uniqueMember values) will be included. The Directory Server provides an entry cache implementation with these settings although it is disabled by default.

To Enable the Entry Cache

- Run dsconfig to enable the entry cache.

```
$ bin/dsconfig set-entry-cache-prop --cache-name "Static Group Entry Cache"
 \
  --set enabled:true
```

To Create Your Own Entry Cache for Large Groups

- You can create your own entry cache for large groups using the dsconfig create-entry-cache subcommand.

```
# bin/dsconfig create-entry-cache --type fifo \
  --set enabled:true \
  --set cache-level:10 \
  --set max-entries:175000 \
  --set "include-filter:(objectClass=groupOfUniqueNames)" \
  --set min-cache-entry-value-count:10000 \
  --set min-cache-entry-attribute:uniquemember
```

Monitoring the Entry Cache

You can monitor the memory consumed by your entry cache using the entry-cache-info property in the periodic stats logger. You can retrieve the monitor entry over LDAP by issuing a search on baseDN="cn=monitor" using filter="(objectClass=ds-fifo-entry-cache-monitor-entry)". For example, the entry might appear as follows:

```
dn: cn=Static Group Entry Cache Monitor,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-fifo-entry-cache-monitor-entry
objectClass: extensibleObject
cn: Static Group Entry Cache Monitor
cacheName: Static Group Entry Cache
entryCacheHits: 6416407
entryCacheTries: 43069073
entryCacheHitRatio: 14
maxEntryCacheSize: 12723879900
currentEntryCacheCount: 1
maxEntryCacheCount: 175000
entriesAddedOrUpdated: 1
evictionsDueToMaxMemory: 0
evictionsDueToMaxEntries: 0
entriesNotAddedAlreadyPresent: 0
entriesNotAddedDueToMaxMemory: 0
entriesNotAddedDueToFilter: 36652665
entriesNotAddedDueToEntrySmallness: 0
lowMemoryOccurrences: 0
percentFullMaxEntries: 0
jvmMemoryMaxPercentThreshold: 75
jvmMemoryCurrentPercentFull: 24
jvmMemoryBelowMaxMemoryPercent: 51
isFull: false
capacityDetails: NOT FULL: The JVM is using 24% of its available memory.
 Entries can be
```

```
added to the cache until the overall JVM memory usage reaches the configured
 limit of
75%. Cache has 174999 remaining entries before reaching the configured limit
 of 175000.
```

By default, the entry cache memory is set to 75%, with a maximum of 90%.

Tuning the Index Entry Limit for Large Groups

The Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the DIT. You can specify an index entry limit property, which defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (which is 4000 by default), then you must rebuild the indexes using the `rebuild-index` tool as follows:

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index objectclass
```

In the majority of Directory Server environments, the default index entry limit value of 4000 entries should be sufficient. However, group-related processing, it may be necessary to increase the index entry limit. For directories containing more than 4000 groups with the same structural object class (i.e., more than 4000 entries, 4000 `groupOfUniqueNames` entries, 4000 `groupOfEntries` entries, or 4000 `groupOfURLs` entries), then you may want to increase the index entry limit for the `objectClass` attribute so that it has a value larger than the maximum number of group entries of each type. Set `index-entry-limit` property using a command line like the following:

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
  --index-name objectClass --set index-entry-limit:175000
```

As an alternative, a separate backend may be created to hold these group entries, so that an unindexed search in that backend yields primarily group entries. If you make no changes, then the internal search performed at startup to identify all groups and any user searches looking for groups of a given type may be very expensive.

For directories in which any single user may be a member of more than 4000 static groups of the same type, you may need to increase the index entry limit for the `member` and/or `uniqueMember` attribute to a value larger than the maximum number of groups in which any user is a member. If you do not increase the limit, then searches to retrieve the set of all static groups in which the user is a member may be unindexed and therefore very expensive.

Summary of Commands to Search for Group Membership

The following summary of commands show the fastest way to retrieve direct or indirect member DNs for groups.

- To retrieve direct member (non-nested) DNs of group "`cn=group.1,ou=groups,dc=example,dc=com`".

  ```
  $ bin/ldapsearch --baseDN "cn=group.1,ou=Groups,dc=example,dc=com"
   "(objectClass=*)" uniqueMember member
  ```
- To retrieve direct member entries (non-nested) under "`dc=example,dc=com`" of group "`cn=group.1,ou=groups,dc=example,dc=com`". This is useful when attributes from member entries are used in the filter or being returned.

  ```
  $ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com"
   "(isDirectMemberOf=cn=group.1,ou=Groups,dc=example,dc=com)"
  ```
- To retrieve group DNs in which user "`uid=user.2,ou=people,dc=example,dc=com`" is a direct member (non-nested, static groups).

  ```
  $ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com"
   "(objectClass=*)" isDirectMemberOf
  ```

- To retrieve all member entries under ou=people,dc=example,dc=com of group "cn=group.1,ou=groups,dc=example,dc=com".

  ```
  $ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com"
   "(isMemberOf=cn=group.1,ou=Groups,dc=example,dc=com)"
  ```
- To retrieve the group DNs in which user "uid=user.2,ou=people,dc=example,dc=com" is a member.

  ```
  $ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com"
   "(objectClass=*)" isMemberOf
  ```

Migrating Sun/Oracle Groups

You can migrate Sun/Oracle static and dynamic groups to PingDirectory Server groups. The following sections outline the procedures for migrating static groups to both Ping Identity static groups and virtual static groups as well as how to migrate dynamic groups. For information about the differences in access control evaluation between Sun/Oracle and the PingDirectory Server, see Migrating ACIs from Sun/Oracle to PingDirectory Server.

Migrating Static Groups

The PingDirectory Server supports static LDAP groups with structural object classes of groupOfNames, groupOfUniqueNames, or groupOfEntries. In general, static groups may be imported without modification.

A FIFO entry cache can be enabled to cache group-to-user mappings, which improves performance when accessing very large entries, though at the expense of greater memory consumption. The PingDirectory Server provides an out-of-the-box FIFO entry cache object for this purpose. This object must be explicitly enabled using dsconfig as described in Using the Entry Cache to Improve the Performace of Large Static Groups.

To Migrate Static Groups

1. Run the migrate-ldap-schema tool to enumerate any schema differences between the DSEE deployment and the Ping Identity deployment.
2. Run the migrate-sun-ds-config tool to enumerate any configuration differences between the DSEE deployment and the Ping Identity deployment.
3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.
4. Import the existing users and groups using the import-ldif tool.
5. From the PingDirectory Server root directory, open the sun-ds-compatibility.dsconfig file in the docs folder using a text editor.
6. Find the **FIFO Entry Cache** section and, after reading the accompanying comments, enable the corresponding dsconfig command by removing the comment character ("#").

   ```
   $ bin/dsconfig set-entry-cache-prop \
     --cache-name "Static Group Entry Cache" --set enabled:true
   ```
7. Enable the Referential Integrity Plug-in. This will ensure that references to an entry are automatically updated when the entry is deleted or renamed.

   ```
   $ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" --set
    enabled:true
   ```

   If this Directory Server is part of a replication topology, you should enable the Referential Integrity Plug-in for each replica.

Migrating Static Groups to Virtual Static Groups

In many cases, electing to use virtual static groups in place of static groups can produce marked performance gains without any need to update client applications. The specifics of a migration to virtual static groups varies depending on the original DIT, but the general approach involves identifying common membership traits for all members of each group and then expressing those traits in the form of an LDAP URL.

In the following example, the common membership trait for all members of the All Users group is the parent DN ou=People,dc=example,dc=com. In other cases, a common attribute may need to be used. For example, groups based on the location of its members could use the `l` (location) or `st` (state) attribute.

To Migrate DSEE Static Groups to Virtual Static Groups

In the following example, consider the common case of an "All Users" group, which contains all entries under the parent DN "`ou=People,dc=example,dc=com`". When implemented as a virtual static group, this group may have a large membership set without incurring the overhead of a static group.

1. First, create a dynamic group.

```
dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Dynamic All Users
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(objectClass=person)
```

2. Next, create a virtual static group that references the dynamic group.

```
dn: cn=All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
objectClass: ds-virtual-static-group
cn: All Users
ds-target-group-dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
```

3. Finally, the `Virtual Static uniqueMember` virtual attribute must be enabled to populate the All Users group with `uniqueMember` virtual attributes.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
 uniqueMember" \
  --set enabled:true
```

4. Confirm that the virtual static group is correctly configured by checking a user's membership in the group.

```
$ bin/ldapsearch --baseDN "cn=All Users,ou=Groups,dc=example,dc=com" \
  --searchScope base "(uniqueMember=uid=user.0,ou=People,dc=example,dc=com)"
 1.1
```

```
dn: cn=All Users,ou=Groups,dc=example,dc=com
```

5. The ability to list all members of a virtual static group is disabled by default. You may enable this feature, but only if specifically required by a client application.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
 uniqueMember" \
  --set allow-retrieving-membership: true
```

Note: The virtual static group may also be implemented using the `groupOfNames` object class instead of `groupOfUniqueNames`. In that case, you must update the `Virtual Static member` configuration object instead of the `Virtual Static uniqueMember` configuration object.

Migrating Dynamic Groups

The PingDirectory Server supports dynamic groups with the `groupofURLs` object class. In general, dynamic groups may be imported without modification.

To Migrate Dynamic Groups

1. Run the `migrate-ldap-schema` tool to enumerate any schema differences between the DSEE deployment and the Ping Identity deployment.
2. Run the `migrate-sun-ds-config` tool to enumerate any configuration differences between the DSEE deployment and the Ping Identity deployment.
3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.
4. Import the existing users and groups using the `import-ldif` tool.

Chapter 14

Encrypting Sensitive Data

Topics:

The Directory Server provides several ways that you can protect sensitive information in the server. If not enabled during server setup, you can enable on-disk encryption for data in backends as well as in the changelog and the replication databases, and you can also protect sensitive attributes by limiting the ways that clients may interact with them.

This chapter presents the following topics:

Encrypting and Protecting Sensitive Data

The Directory Server provides an encryption-settings database that holds encryption and decryption definitions to protect sensitive data. You can enable on-disk encryption for data in backends as well as in the changelog and the replication databases. You can also protect sensitive attributes by limiting the ways that clients may interact with them.

About the Encryption-Settings Database

The encryption-settings database is a repository that the server uses to hold information for encrypting and decrypting data. The database contains any number of *encryption-settings definitions* that specifies information about the cipher transformation and encapsulates the key used for encryption and decryption.

Before data encryption can be enabled, you first need to create an encryption-settings definition. An encryption-settings definition specifies the cipher transformation that should be used to encrypt the data, and encapsulates the encryption key. The `encryption-settings` command-line tool can be used to manage the encryption settings database, including creating, deleting, exporting, and importing encryption-settings definitions, listing the available definitions, and indicating which definition should be used for subsequent encryption operations.

Although the encryption-settings database can have multiple encryption-settings definitions, only one of them can be designated as the *preferred* definition. The preferred encryption-settings definition is the one that will be used for any subsequent encryption operations. Any existing data that has not yet been encrypted remains unencrypted until it is rewritten (e.g., as a result of a modify or modify DN operation, or if the data is exported to LDIF and re-imported). Similarly, if you introduce a new preferred encryption-settings definition, then any existing encrypted data will continue to use the previous definition until it is rewritten. If you do change the preferred encryption-settings definition for the server, then it is important to retain the previous definitions until you are confident that no remaining data uses those older keys.

Supported Encryption Ciphers and Transformations

The set of encryption ciphers that are supported by the Directory Server is limited to those ciphers supported by the JVM in which the server is running. For specific reference information about the algorithms and transformations available in all compliant JVM implementations, see the following:

- Java Cryptography Architecture Reference Guide
- Java Cryptography Architecture Standard Algorithm Name Documentation

When configuring encryption, the cipher to be used must be specified using a key length (in bits) and either a cipher algorithm name (e.g., "AES") or a full cipher transformation which explicitly specifies the mode and padding to use for the encryption (e.g., "AES/CBC/ PKCS5Padding"). If only a cipher algorithm is given, then the default mode and padding for that algorithm will be automatically selected.

The following cipher algorithms and key lengths have been tested using the Sun/Oracle JVM:

Table 26: Cipher Algorithms

Cipher Algorithm	Key Length (bits)
AES	128
Blowfish	128
DES	64
DESede	192
RC4	128

Note: By default, some JVM implementations may come with limited encryption strength, which may restrict the key lengths that can be used. For example, the Sun/Oracle JVM does not allow AES with 192-bit or 256-bit keys unless the unlimited encryption strength policy files are downloaded and installed.

The Directory Server supports four Cipher Stream Providers, which are used to obtain cipher input and output streams to read and write encrypted data.

Table 27: Cipher Stream Providers

Cipher Stream Providers	Description
Default	Default cipher stream provider using a hard-coded default key.
File-Based	Used to read a specified file in order to obtain a password used to generate cipher streams for reading and writing encrypted data.
Third-Party	Used to provide cipher stream provider implementations created in third-party code using the Server SDK.
Wait-for-Passphrase	Causes the server to wait for an administrator to enter a passphrase that will be used to derive the key for cipher streams. You can supply the passphrase to the server by running `encryption-settings supply-passphrase`.

Using the encryptions-settings Tool

The `encryption-settings` tool provides a mechanism for interacting with the server's encryption-settings database. It may be used to list the available definitions, create new definitions, delete existing definitions, and indicate which definition should be the preferred definition. It may also be used to export definitions to a file for backup purposes and to allow them to be imported for use in other Directory Server instances.

To List the Available Encryption Definitions

- Use the `encryption-settings` tool with the `list` subcommand to display the set of available encryption settings definitions. This subcommand does not take any arguments. For each definition, it will include the unique identifier for the definition, as well as the cipher transformation and key length that will be used for encryption and whether it is the preferred definition.

```
$ bin/encryption-settings list
```

```
Encryption Settings Definition ID: 4D86C7922F71BB57B8B5695D2993059A26B8FC01
Preferred for New Encryption: false
Cipher Transformation: DESede
Key Length (bits): 192

Encryption Settings Definition ID: F635E109A8549651025D01D9A6A90F7C9017C66D
Preferred for New Encryption: true
Cipher Transformation: AES
Key Length (bits): 128
```

Creating Encryption-Settings Definitions

To create a new encryption-settings definition, use the create subcommand. This subcommand takes the following arguments:

- **--cipher-algorithm {algorithm}**. Specifies the base cipher algorithm that should be used. This should just be the name of the algorithm (e.g., "AES", "DES", "DESede", "Blowfish", "RC4", etc.). This argument is required.
- **--cipher-transformation {transformation}**. Specifies the full cipher transformation that should be used, including the cipher mode and padding algorithms (e.g., "AES/CBC/ PKCS5Padding"). This argument is optional, and if it is not provided, then the JVM-default transformation will be used for the specified cipher algorithm.

- **--key-length-bits {length}**. Specifies the length of the encryption key in bits (e.g., 128). This argument is required.
- **--set-preferred**. Indicates that the new encryption-settings definition should be made the preferred definition and therefore should be used for subsequent encryption operations in the server. When creating the first definition in the encryption-settings database, it will automatically be made the preferred definition.

To Create an Encryption-Settings Definition

- Use the `encryption-settings` tool with the create subcommand to specify the definition.

```
$ bin/encryption-settings create --cipher-algorithm AES \
  --key-length-bits 128 --set-preferred
```

```
Successfully created a new encryption settings definition with ID
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Changing the Preferred Encryption-Settings Definition

To change the preferred encryption-settings definition, use the `encryption-settings` tool with the `set-preferred` subcommand. This subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument is required.

To Change the Preferred Encryption-Settings Definition

- Use the `encryption-settings` tool with the set-preferred subcommand to change a definition to a preferred definition.

```
$ bin/encryption-settings set-preferred --id
 4D86C7922F71BB57B8B5695D2993059A26B8FC01
```

```
Encryption settings definition 4D86C7922F71BB57B8B5695D2993059A26B8FC01 was
successfully set as the preferred definition for subsequent encryption
 operations
```

Deleting an Encryption-Settings Definition

To delete an encryption-settings definition, use the `encryption-settings` tool with the delete subcommand. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be deleted. This argument is required.

Never delete an encryption-settings definition if data in the server is still encrypted using the settings contained in that definition. Any data still encrypted with a definition that has been removed from the database will be inaccessible to the server and will cause errors for any attempt to access it. This includes the replicationChanges and changelog databases, which the `re-encode-entries` tool will not re-encode with the new encryption-settings definition. Therefore, wait for the amount of time defined in the `replication-purge-delay`, of the Replication Server, and `changelog-maximum-age` of the changelog Backend (if enabled) before removing previous encryption-settings definitions. To safely delete a compromised encryption-settings definition, see the the Dealing with a Compromised EncryptionKey section.

To stop using a definition for encryption and use a different definition, make sure that the desired definition exists in the encryption-settings database and set it to be the preferred definition. As long as the encryption key has not been compromised, there is no harm in having old encryption-settings definitions available to the server, and it is recommended that they be retained just in case they are referenced by something.

The preferred encryption-settings definition cannot be deleted unless it is the only one left. To delete the currently-preferred definition when one or more other definitions are available, make one of the other definitions preferred as described in the previous section.

To Delete an Encryption-Settings Definition

- Use the `encryption-settings` command with the delete subcommand. Make sure to include the --id argument to specify the definition.

```
$ bin/encryption-settings delete --id
 F635E109A8549651025D01D9A6A90F7C9017C66D
```

```
Successfully deleted encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Configuring the Encryption-Settings Database

Because the encryption-settings database contains the encryption keys used to protect server data, the contents of the encryption-settings database is itself encrypted. By default, the server will derive a key to use for this purpose, but it is recommended that you customize the logic used to access the encryption-settings database with a cipher stream provider. The Server SDK provides an API that can be used to create custom cipher stream provider implementations, but the server also comes with one that will obtain the key from a PIN file that you create (see the example procedure below).

To Configure the Encryption-Settings Database

1. Use `dsconfig` to configure the server so that the encryption-settings database is encrypted with a PIN contained in the file `config/encryption-settings.pin`.

```
$ bin/dsconfig create-cipher-stream-provider \
  --provider-name "Encryption Settings PIN File" \
  --type file-based \
  --set enabled:true \
  --set password-file:config/encryption-settings.pin
```

2. Use `dsconfig` to set the global configuration property for the cipher stream provider, which sets the on-disk encryption.

```
$ bin/dsconfig set-global-configuration-prop \
  --set "encryption-settings-cipher-stream-provider:Encryption Settings PIN
 File"
```

3. Use the `encryption-settings` tool to create a new encryption-settings definition. This command automatically generates a new 256-bit encryption key for use with AES encryption, and mark it as the preferred definition for future encryption operations in the server. Note that this command will fail if you do not have the unlimited encryption strength policy installed as described in the previous section (if you do not have that policy installed, then you are restricted to a 128-bit key for AES encryption).

```
$ bin/encryption-settings create \
  --cipher-algorithm AES \
  --key-length-bits 256 \
  --set-preferred
```

4. Obtain a list of the definitions in the encryption-settings database.

```
$ bin/encryption-settings list
```

5. You can export an encryption-settings definition from the database using a command like the following where the encryption-settings ID should be changed as necessary to suit your deployment:

```
$ bin/encryption-settings export \
  --id DA39A3EE5E6B4B0D3255BFEF95601890AFD80709 \
  --output-file /tmp/exported-key \
  --pin-file /tmp/exported-key.pin
```

6. If no PIN file is specified, then you will be interactively prompted to provide it. To import an encryption-settings definition into the database on another server.

```
$ bin/encryption-settings import \
  --input-file /tmp/exported-key \
  --pin-file /tmp/exported-key.pin \
  --set-preferred
```

Backing Up and Restoring the Encryption-Settings Definitions

If using data encryption in a Directory Server instance, do not lose the encryption-settings definitions used to encrypt data in the server. If an encryption-settings definition is lost, any data encrypted with that definition will be completely inaccessible. Make sure the encryption-settings definitions are backed up regularly.

The Directory Server provides two different mechanisms for backing up and restoring encryption-settings definitions. One or more encryption-settings definitions can be exported and imported using the `encryption-settings` tool. Or, the entire encryption-settings database can be backed up and restored using the Directory Server's `backup` and `restore` tools.

If a pin file is used to define a passphrase to the encryption-settings database, the passphrase must be backed up and kept secure independently of the userRoot and encryption-settings database backups. The passphrase in the pin file is needed if the encryption-settings database is to be restored into a different server root.

Exporting Encryption-Settings Definitions

To back up an individual definition (or to export it from one server so that you can import it into another), use the export subcommand to the `encryption-settings` command. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument can be specified multiple times. If it is omitted, all definitions are exported.
- **--output-file {path}**. Specifies the path to the output file to which the encryption-settings definition will be written. This argument is required.
- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.

To Export an Encryption-Settings Definition

- Use the `encryption-settings` tool with the export subcommand to export the definition to a file.

```
$ bin/encryption-settings export --id
 F635E109A8549651025D01D9A6A90F7C9017C66D \
  --output-file /tmp/exported-key
Enter the PIN to use to encrypt the definition:
Re-enter the encryption PIN:
```

```
Successfully exported encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D to file /tmp/exported-key
```

Importing Encryption-Settings Definitions

To import an encryption-settings definition that has been previously exported, use the `encryption-settings` tool with the import subcommand. The subcommand takes the following arguments:

- **--input-file {path}**. Specifies the path to the file containing the exported encryption-settings definition. This argument is required.
- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.
- **--set-preferred**. Specifies that the newly-imported encryption-settings definition should be made for the preferred definition for subsequent encryption-settings.

To Import an Encryption-Settings Definition

- Use the `encryption-settings` tool with the import subcommand to import the definition to a file.

```
$ bin/encryption-settings import --input-file /tmp/exported-key --set-
preferred
Enter the PIN used to encrypt the definition:
```

```
Successfully imported encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D from file /tmp/exported-key
```

Enabling Data Encryption in the Server

Encryption can be enabled during server setup, by defining an encryption key and passphrase. This configuration should be used across all servers in a topology. On legacy systems or post setup, data encryption can be configured by having at least one encryption-settings definition available for use. Then, set the value of the `encrypt-data` global configuration property to true.

Setting the global configuration property will automatically enable data encryption for all types of backends that support it (including the changelog backend and indexes), as well as for the replication server database. All subsequent write operations will cause the corresponding records written into any of these locations to be encrypted. Any existing data will remain unencrypted until it is rewritten by a write operation. If you wish to have existing data encrypted, then you will need to export that data to LDIF and re-import it. This will work for both the data backends, the changelog, and indexes, but it is not an option for the replication database, so existing change records will remain unencrypted until they are purged. If this is not considered acceptable in your environment, then follow the steps in the Dealing with a Compromised Encryption Key to safely purge the replication database.

Configuration for backups and LDIF exports can be done with the following global properties:

- `automatically-compress-encrypted-ldif-exports`. Indicates whether to automatically compress LDIF exports that are also encrypted. If set to true, any LDIF export that is encrypted (either explicitly with `--encryptLDIF`,or implicitly with the `encrypt-ldif-exports-by-default` configuration property) will automatically be gzip-compressed. If this is false, encrypted LDIF exports can still be manually compressed using the `--compress` command-line argument.
- `backup-encryption-settings-definition-id`. The unique identifier for the encryption settings definition to use to generate the encryption key for encrypted backups by default. If this property is given a value, then a definition with that ID must exist in the server's encryption settings database. If this property is not given a value, but the server is configured with at least one encryption settings definition, then the preferred definition is used. If no encryption settings definitions are available, the server will use an internal key shared among servers in the topology. Regardless of this property's value, it can be overridden with the `backup` command-line tool. Providing one of the `--promptForEncryptionPassphrase` or `--encryptionPassphraseFile` arguments will generate the encryption key from the provided passphrase. Or, the `--encryptionSettingsDefinitionID` argument can be used to generate the key from the specified encryption settings definition.
- `encrypt-backups-by-default`. Indicates whether the server should encrypt backups by default. If set to true, a defined `backup-encryption-settings-definition-id` value will be used to generate the encryption key for the backup. If this property is true, and if a `backup-encryption-settings-definition-id` value is not specified, the server will try to use the preferred encryption settings definition to generate the encryption key. If the server is not configured with any encryption settings definitions, an internal key that is shared among instances in the topology is used. Regardless of this property's value, it can be overridden with the `backup` command-line tool's `--encrypt` argument, even if this property is set to false. The `--doNotEncrypt` argument will always cause the backup to be unencrypted, even if this property has a value of true.
- `encrypt-ldif-exports-by-default`. Indicates whether the server should encrypt LDIF exports by default. If set to true, and an `ldif-export-encryption-settings-definition-id` value is specified, then that encryption settings definition is used to generate the encryption key for the export. If this property is true, and an `ldif-export-encryption-settings-definition-id` value is not specified, the server will first try to use the preferred encryption settings definition to generate the encryption key. If the server is not configured with any encryption settings definitions, then an internal key that is shared among

instances in the topology is used. Regardless of this property's value, the default behavior can be overridden with the `export-ldif` command-line tool. The tool's `--encryptLDIF` argument will always encrypt the export, and the `--doNotEncryptLDIF` argument will always create an unencrypted export.

To Enable Data Encryption in the Server

- Use `dsconfig` to set the global configuration property for data encryption to true.

```
$ bin/dsconfig set-global-configuration-prop --set encrypt-data:true
```

Using Data Encryption in a Replicated Environment

Data encryption is only used for the on-disk storage for data within the server. Whenever clients access that data, it is presented in unencrypted form (although the communication with those clients may itself be encrypted using SSL or StartTLS). Replication, the communication of updates between replication servers, is always encrypted using SSL. Each server may apply data encryption in a completely independent manner and have different sets of encryption-settings definitions. It is also possible to have a replication topology containing some servers with data encryption enabled and others with it disabled.

However, when initializing the backend of one server from another server with data encryption enabled, then the server being initialized must have access to all encryption-settings definitions that may have been used for data contained in that backend. To do this, perform an export of the encryption-settings database on the source server using `bin/encryption-settings export` and import it on the target server using `bin/encryption-settings import`.

Dealing with a Compromised Encryption Key

If an encryption-settings definition becomes compromised such that an unauthorized individual obtains access to the encryption key, then any data encrypted with that definition is also vulnerable because it can be decrypted using that key. It is very important that the encryption-settings database be protected (e.g., using file permissions and/or filesystem ACLs) to ensure that its contents remain secure.

In the event that an encryption-settings definition is compromised, then you should immediately stop using that definition. Any data encrypted with the compromised key should be re-encrypted with a new definition or purged from the server. This can be done on one server at a time to avoid an environment-wide downtime, but it should be completed as quickly as possible on all servers that had used that definition at any point in the past in order to minimize the risk of that data becoming exposed.

To Deal with a Compromised Encryption Key

The recommended process for responding to a compromised encryption settings definition is as follows:

1. Create a new encryption-settings definition and make it the preferred definition for new writes.
2. Ensure that client traffic is routed away from the server instance to be updated. For example, if the Directory Server is accessed through a Directory Proxy Server, then you may set the `health-check-state` configuration property for any LDAP external server definitions that reference that server to have a value of unavailable.
3. Ensure that external clients are not allowed to write operations in the directory server instance. This may be accomplished by setting the `writability-mode` global configuration property to have a value of `internal-only`.
4. Wait for all outstanding local changes to be replicated out to other servers. This can be accomplished by looking at the monitor entries with the `ds-replication-server-handler-monitor-entry` object class to ensure that the value of the `update-sent` attribute is no longer increasing.
5. Stop the directory server instance.
6. Delete the replication server database by removing all files in the `changeLogDb` directory below the server root. As long as all local changes have been replicated out to other servers, this will not result in any data loss in the replication environment.

7. Export the contents of all local DB and changelog backends to LDIF. Then, re-import the data from LDIF, which will cause it to be encrypted using the new preferred encryption settings definition.
8. Export the compromised key from the encryption settings database to back it up in case it may be needed again in the future (e.g., if some remaining data was later found to have been encrypted with the key contained in that definition). Then, delete it from the encryption settings database so that it can no longer be used by that directory server instance.
9. Start the directory server instance.
10. Allow replication to bring the server back up-to-date with any changes processed while it was offline.
11. Re-allow externally-initiated write operations by changing the value of the global `writability-mode` configuration property back to enabled.
12. Re-configure the environment to allow client traffic to again be routed to that server instance (e.g., by changing the value of the "health-check-state" property in the corresponding LDAP external instance definitions in the Directory Proxy Server instances back to "dynamically-determined").

Configuring Sensitive Attributes

Data encryption is only applied to the on-disk storage for a Directory Server instance. It does not automatically protect information accessed or replicated between servers, although the server offers other mechanisms to provide that protection (i.e., SSL, StartTLS, SASL). Ensuring that all client communication uses either SSL or StartTLS encryption and ensuring that all replication traffic uses SSL encryption ensures that the data is protected from unauthorized individuals who may be able to eavesdrop on network communication. This communication security may be enabled independently of data encryption (although if data encryption is enabled, then it is strongly recommended that secure communication be used to protect network access to that data).

However, for client data access, it may not be as simple as merely enabling secure communication. In some cases, it may be desirable to allow insecure access to some data. In other cases, it may be useful to have additional levels of protection in place to ensure that some attributes are even more carefully protected. These kinds of protection may be achieved using sensitive attribute definitions.

Each sensitive attribute definition contains a number of configuration properties, including:

- **`attribute-type`**. Specifies the set of attribute types whose values may be considered sensitive. At least one attribute type must be provided, and all specified attribute types must be defined in the server schema.
- **`include-default-sensitive-operational-attributes`**. Indicates whether the set of sensitive attributes should automatically be updated to include any operational attributes maintained by the Directory Server itself that may contain sensitive information. At present, this includes the `ds-sync-hist` operation attribute, which is used for data required for replication conflict resolution and may contain values from other attributes in the entry.
- **`allow-in-filter`**. Indicates whether sensitive attributes may be used in filters. This applies not only to the filter used in search requests, but also filters that may be used in other places, like the assertion and join request controls. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be used in filters over both secure and insecure connections)
  - `Reject` (reject any request which includes a filter targeting one or more sensitive attributes over both secure and insecure connections)
  - `Secure-only` (allow sensitive attributes to be used in filters over secure connections, but reject any such requests over insecure connections)
- **`allow-in-add`**. Indicates whether sensitive attributes may be included in entries created by LDAP add operations. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be included in add requests over both secure and insecure connections)
  - `Reject` (reject any add request containing sensitive attributes over both secure and insecure connections)
  - `Secure-only` (allow sensitive attributes to be included in add requests received over a secure connection, but reject any such requests over an insecure connection)

- **`allow-in-compare`**. Indicates whether sensitive attributes may be targeted by the assertion used in a compare operation. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be targeted by requests over both secure and insecure connections)
  - `Reject` (reject any compare request targeting a sensitive attribute over both secure and insecure connections)
  - `Secure-only` (allow compare requests targeting sensitive attributes over a secure connection, but reject any such requests over an insecure connection)
- **`allow-in-modify`**. Indicates whether sensitive attributes may be updated using modify operations. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be modified by requests over both secure and insecure connections)
  - `Reject` (reject any modify request updating a sensitive attribute over both secure and insecure connections)
  - `Secure-only` (only modify requests updating sensitive attributes over a secure connection, but reject any such request over an insecure connection)

The `allow-in-returned-entries`, `allow-in-filter`, `allow-in-add`, `allow-in-compare`, and `allow-in-modify` properties all have default values of `secure-only`, which prevents the possibility of exposing sensitive data in the clear to anyone able to observe network communication.

If a client connection policy references a sensitive attribute definition, then any restrictions imposed by that definition will be enforced for any clients associated with that client connection policy. If multiple sensitive attribute definitions are associated with a client connection policy, then the server will use the most restrictive combination of all of those sets.

Note that sensitive attribute definitions work in conjunction with other security mechanisms defined in the server and may only be used to enforce additional restrictions on clients. Sensitive attribute definitions may never be used to grant a client additional access to information that it would not have already had through other means. For example, if the `employeeSSN` attribute is declared to be a sensitive attribute and the `allow-in-returned-entries` property has a value of `Secure-only`, then the `employeeSSN` attribute will only be returned to those clients that have both been granted permission by the access control rules defined in the server and are communicating with the server over a secure connection. The `employeeSSN` attribute will be stripped out of entries returned to clients normally authorized to see it if they are using insecure connections, and it will also be stripped out of entries for clients normally not authorized to see it even if they have established secure connections.

To Create a Sensitive Attribute

1. To create a sensitive attribute, you must first create one or more sensitive attribute definitions.

   For example, to create a sensitive attribute definition that will only allow access to the `employeeSSN` attribute by clients using secure connections, the following configuration changes may be made:

```
$ bin/dsconfig create-sensitive-attribute \
  --attribute-name "Employee Social Security Numbers" \
  --set attribute-type:employeeSSN \
  --set include-default-sensitive-operational-attributes:true \
  --set allow-in-returned-entries:secure-only \
  --set allow-in-filter:secure-only \
  --set allow-in-add:secure-only \
  --set allow-in-compare:secure-only \
  --set allow-in-modify:secure-only
```

2. Associate those sensitive attribute definitions with the client connection policies for which you want them to be enforced.

```
$ bin/dsconfig set-client-connection-policy-prop --policy-name default \
  --set "sensitive-attribute:Employee Social Security Numbers"
```

Configuring Global Sensitive Attributes

Administrators can assign one or more sensitive attribute definitions to a client connection policy. However, in an environment with multiple client connection policies, it could be easy to add a sensitive attribute definition to one policy but overlook it in another. The Directory Server supports the ability to define sensitive attributes as a global configuration option so that they will automatically be used across all client connection policies.

To Configure a Global Sensitive Attribute

- Run `dsconfig` to add a global sensitive attribute across all client connection policies. The following command adds the `employeeSSN` as a global sensitive attribute, which is applied across all client connection policies.

```
$ bin/dsconfig set-global-configuration-prop --add "sensitive-
attribute:employeeSSN"
```

Excluding a Global Sensitive Attribute on a Client Connection Policy

Administrators can set a global sensitive attribute across all client connection policies. However, there may be cases when a specific directory server must exclude the sensitive attribute as it may not be needed for client connection requests. For example, in most environments it is good to declare the `userPassword` attribute to be a sensitive attribute in a manner that prevents it from being read by external clients. Further, this solution is more secure than protecting the `password` attribute using the server's default global ACI, which only exists for backwards compatibility purposes. If the Data Sync Server is installed, then it does need to be able to access passwords for synchronization purposes. In this case, the administrator can set `userPassword` to be a sensitive attribute in all client connection policies, but exclude it in a policy specifically created for use by the Data Sync Server. The Directory Server provides an `exclude-global-sensitive-attribute` property for this purpose.

To Exclude a Global Sensitive Attribute on a Client Connection Policy

1. Run `dsconfig` to remove the global ACI that limits access to the `userPassword` or `authPassword` attribute. This is present for backwards compatibility.

```
$ bin/dsconfig set-access-control-handler-prop \
  --remove 'global-aci:(targetattr="userPassword || authPassword")
   (version 3.0; acl "Prevent clients from retrieving passwords from the
 server";
  deny (read,search,compare) userdn="ldap:///anyone";)'
```

2. Run `dsconfig` to add the `userPassword` attribute as a global sensitive attribute, which is applied to all client connection policies. Do this by adding the built-in "Sensitive Password Attributes" Sensitive Attribute definition to the Global Configuration.

```
$ bin/dsconfig set-global-configuration-prop \
  --add "sensitive-attribute:Sensitive Password Attributes"
```

3. If the server is designated to synchronize passwords with a Sync Server, then it is necessary to configure a client connection policy for the Sync User to exclude the global sensitive attribute. The following is an example on how to create a new policy if the Data Sync Server binds with the default DN of `cn=Sync User,cn=Root DNs,cn=config`.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Requests by Sync Users" \
  --type simple \
  --set user-auth-type:internal \
  --set user-auth-type:sasl \
  --set user-auth-type:simple \
  --set "included-user-base-dn:cn=Sync User,cn=Root DNs,cn=config"
```

```
$ bin/dsconfig create-client-connection-policy \
  --policy-name "Data Sync Server Connection Policy" \
  --set enabled:true \
  --set evaluation-order-index:9998 \
  --set "connection-criteria:Requests by Sync Users" \
  --set "exclude-global-sensitive-attribute:Sensitive Password Attributes"
```

Chapter 15

Working with the LDAP Changelog

Topics:

The Directory Server provides a client-accessible LDAP changelog (based on the Changelog Internet Draft Specification) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services.

This chapter will present the following topics related to the LDAP Changelog:

Overview of the LDAP Changelog

The Directory Server provides a client-accessible LDAP changelog (based on the Changelog Internet Draft Specification) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services. For example, you can install the Data Sync Server that monitors the LDAP changelog for any updates that occur on a source directory server and synchronizes these changes to a target DIT or database server. The Directory Server provides an additional feature in that the LDAP changelog supports virtual attributes.

👉 **Note:** The LDAP Changelog should not be confused with the Replication Changelog. The main distinction is as follows:

- The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at `<server-root>/db/changelog`.
- The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at `<server-root>/changelogDB`.

Key Changelog Features

As of version 3.2, the Directory Server supports two new Changelog Backend properties that allow access control filtering and sensitive attribute evaluation for targeted entries. External client applications can change the contents of attributes they can see in the targeted entry based on the access control rules applied to the associated base DN.

- **apply-access-controls-to-changelog-entry-contents**. Indicates whether the contents of changelog entry attributes (i.e., `changes`, `deletedEntryAttrs`, `ds-changelog-entry-key-attr-values`, `ds-changelog-before-values`, and `ds-changelog-after-values`) are subject to access control and/ or sensitive attribute evaluation to limit data that LDAP clients can see. The client must have the access control permissions to read changelog entries to retrieve them in any form. If this feature is enabled and the client does not have permission to read an entry at all, or if that client does not have permission to see any attributes that were targeted by the change, then the associated changelog entries targeted by those operations will be suppressed. If a client does not have permission to see certain attributes within the target entry, then references to those attributes in the changelog entry will also be suppressed. This property only applies to standard LDAP searches of the `cn=changelog` branch.
- **report-excluded-changelog-attributes**. Indicates whether to include additional information about any attributes that may have been removed due to access control filtering. This property only applies to content removed as a result of processing performed by the `apply-access-controls-to-changelog-entry-contents` property. Possible values are:
  - **none** - Indicates that changelog entries should not include any information about attributes that have been removed.
  - **attribute-counts** - Indicates that changelog entries should include a count of user and/ or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, the number of the excluded user attributes will be reported in the `ds-changelog-num-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the number of the excluded operational attributes will be reported in the `ds-changelog-num-excluded-operational-attributes` attribute of the changelog entry. Both the `ds-changelog-num-excluded-user-attributes` and `ds-changelog-num-excluded-operational-attributes` are operational and must be explicitly requested by clients (or all operational attributes requested using "+") to be returned.
  - **attribute-names** - Indicates that changelog entries should include the names of user and/or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, then the names of the excluded user attributes will be reported in the `ds-changelog-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the names of the excluded operational attributes will be reported in the `ds-change-log-excluded-operational-attribute` attribute of the changelog entry. Both the `ds-changelog-excluded-user-attribute` and `ds-changelog-excluded-operational-`

`attribute` attributes are operational and must be explicitly requested by clients (or all operational attributes requested via "+") to be returned.

To Enable Access Control Filtering in the LDAP Changelog

To set up access control to the LDAP Changelog, use the `dsconfig` tool to enable the properties to the Changelog Backend. Only admin users with the `bypass-acl` privilege can read the changelog.

1. Enable the `apply-access-control-to-changelog-entry-contents` property to allow LDAP clients to undergo access control filtering using standard LDAP searches of the `cn=changelog` backend.

   ```
   $ bin/dsconfig set-backend-prop --backend-name "changelog" \
     --set "apply-access-controls-to-changelog-entry-contents:true"
   ```

   Access control filtering will be applied regardless of the value of the `apply-access-controls-to-changelog-entry-contents` setting when the Changelog Backend is servicing requests from an PingDataSync Server that has the `filter-changes-by-user` Sync Pipe property set.
2. Optional. Set the `report-excluded-changelog-attributes` property to include a count of users that have been removed through access control filtering. The count appears in the `ds-changelog-num-excluded-user-attributes` attribute for users and the `ds-changelog-num-excluded-operational-attributes` attribute for operational attributes.

   ```
    $ bin/dsconfig set-backend-prop --backend-name "changelog" \
     --set "report-excluded-changelog-attributes:attribute-counts"
   ```

Useful Changelog Features

The Directory Server provides two useful changelog configuration properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

- **changelog-max-before-after-values**. Setting this property to a non-zero value causes all of the old values and all of the new values (up to the specified maximum) for each changed attribute to be stored in the changelog entry. The values will be stored in the `ds-change-log-before-values` and `ds-changelog-after-values` attributes on the changelog entry. These attributes are not present by default.

  Note: The `changelog-max-before-after-values` property can be expensive for attributes with hundreds or thousands of values, such as a group entry.

  If any attribute has more than the maximum number of values, their names and number of before/after values will be stored in the `ds-changelog-attr-exceeded-max-values-count` attribute on the changelog entry. This is a multi-valued attribute whose format is:

  ```
  attr=attributeName,beforeCount=100,afterCount=101
  ```

  where "attributeName" is the name of the attribute and the "beforeCount" and "afterCount" are the total number of values for that attribute before and after the change, respectively. This attribute indicates that you need to reset the `changelog-max-before-after-values` property to a higher value. When this attribute is set, an alert will be generated.

  Note: If the number of values for an attribute exceeds the maximum value set by the `changelog-max-before-after-values` property, then those values will not be stored.
- **changelog-include-key-attribute**. This property is used for correlation attributes that need to be synchronized across servers, such as `uid`. It causes the current (after-change) value of the specified attributes to be recorded in the `ds-changelog-entry-key-attr-values` attribute on the changelog entry. This applies for all change types. On a DELETE operation, the values are from the entry before they were deleted.

  The key values will be recorded on every change and override the settings configured in `changelog-include-attribute`, `changelog-exclude-attribute`, `changelog-deleted-entry-include-attribute`, or `changelog-deleted-entry-exclude-attribute`.

Example of the Changelog Features

After the `changelog-max-before-after-values` property is set, the before-and-after values of any change attribute will be recorded in the LDAP Changelog. For example, given a simple entry with two multi-valued mail attributes:

```
dn: uid=test,dc=example,dc=com
objectclass: inetorgperson
cn: test user
sn: user
description: oldDescription
mail: test@yahoo.com
mail: test@gmail.com
```

Then, apply the following changes to the entry:

```
dn: uid=test,dc=example,dc=com
changetype: modify
add: mail
mail: test@hotmail.com
-
delete: mail
mail: test@yahoo.com
-
replace: description
description: newDescription
```

The resulting changelog would record the following attribute values:

```
dn: changeNumber=1,cn=changelog
objectClass: top
objectClass: changeLogEntry
targetDN: uid=test,dc=example,dc=com
changeType: modify
changes::
YWRkOiBtYWlsCm1haWw6IHRlc3RAaG90bWFpbC5jb20KLQpkZWxldGU6IG1haWwKbWFpbDogdGVzdEB5YWh
vby5jb20KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogbmV3RGVzY3JpcHRpb24KLQpyZX
BsYWNlOiBtb2RpZmllcnNOYW1lCm1vZGlmaWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VyLGNuPVJvb
3QgRE5zLGNuPWNvbmZpZwotCnJlcGxhY2U6IGRzLXVwZGF0ZS10aW1lCmRzLXVwZGF0ZS10aW1lOjogQUFB
QkxxQitIaTQ9Ci0KAA==
ds-changelog-before-values::
 ZGVzY3JpcHRpb246IG9sZERlc2NyaXB0aW9uCm1haWw6IHRlc3RAeW
Fob28uY29tCm1haWw6IHRlc3RAZ21haWwuY29tCmRzLXVwZGF0ZS10aW1lOjogQUFBQkxxQjdaZ1E9Cm1vZ
GlmaWVyc05hbWU6IGNuPURpcmVjdG 9yeSBNYW5hZ2VyLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwo=
ds-changelog-after-values::
 ZGVzY3JpcHRpb246IG5ld0Rlc2NyaXB0aW9uCm1haWw6IHRlc3RAZ21
haWwuY29tCm1haWw6IHRlc3RAaG90bWFpbC5jb20KZHMtdXBkYXRlLXRpbWU6OiBBQUFCTHFCK0hpND0KbW
9kaWZpZXJzTmFtZTogY249RGlyZWN0b3J5IE1hbmFnZXIsY249Um9vdCBETnMsY249Y29uZmlnCg==
ds-changelog-entry-key-attr-values:: dWlkOiB0ZXN0Cg==
changenumber: 1
```

You can run the `bin/base64` decode -d command-line tool to view the decoded value for the changes, `ds-changelog-before-values`, `ds-changelog-after-values` attributes:

After base64 decoding, the `changes` attribute reads:

```
add: mail
mail: test@hotmail.com
-
delete: mail
mail: test@yahoo.com
-
replace: description
```

```
description: newDescription
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
-
```

After base64 decoding, the `ds-changelog-before-values` attribute reads:

```
description: oldDescription
mail: test@yahoo.com
mail: test@gmail.com
modifyTimestamp: 20131010020345.546Z
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

After base64 decoding, the `ds-changelog-after-values` attribute reads:

```
description: newDescription
mail: test@gmail.com
mail: test@hotmail.com
modifyTimestamp: 20131010020345.546Z
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

Viewing the LDAP Changelog Properties

You can view the LDAP changelog properties by running the `dsconfig` get-backend-prop command and specifying the changelog backend.

To View the LDAP Changelog Properties Using dsconfig Non-Interactive Mode

- Use `dsconfig` to view the changelog properties on the Directory Server. To view "advanced" properties that are normally hidden, add the --advanced option when running the command. For a specific description of each property, see the *PingDirectory Server Configuration Reference*.

```
$ bin/dsconfig get-backend-prop --backend-name changelog
```

```
Property                                     : Value(s)
------------------------------------------:-------------
backend-id                                   : changelog
description                                  : -
enabled                                      : false
writability-mode                             : disabled
base-dn                                      : cn=changelog
set-degraded-alert-when-disabled             : false
return-unavailable-when-disabled             : false
db-directory                                 : db
db-directory-permissions                     : 700
changelog-maximum-age                        : 2 d
db-cache-percent                             : 1
changelog-include-attribute                  : -
changelog-exclude-attribute                  : -
changelog-deleted-entry-include-attribute : -
changelog-deleted-entry-exclude-attribute : -
changelog-include-key-attribute              : -
changelog-max-before-after-values            : 0
changelog-write-batch-size                   : 100
changelog-purge-batch-size                   : 1000
changelog-write-queue-capacity               : 100
write-lastmod-attributes                     : true
```

```
use-reversible-form                          : false
je-property                                  : -
```

Enabling the LDAP Changelog

By default, the LDAP changelog is disabled on the Directory Server. If you are using the `dsconfig` tool in interactive mode, the changelog appears in the Backend configuration as a Standard object menu item.

Note: You can enable the feature using the `dsconfig` tool only if required as it can significantly affect LDAP update performance.

To Enable the LDAP Changelog Using dsconfig Non-Interactive Mode

- Use `dsconfig` to enable the changelog property on the Directory Server.

```
$ bin/dsconfig set-backend-prop \
  --backend-name changelog --set enabled:true
```

To Enable the LDAP Changelog Using Interactive Mode

1. Use `dsconfig` to enable the changelog on each server in the network. Then, authenticate to the server by entering the host name, LDAP connection, port, bindDN and bind password.

```
$ bin/dsconfig
```

2. On the Directory Server main menu, type `o` to change from the Basic object level to the Standard object level.
3. Enter the option to select the Standard Object level.
4. On the Directory Server main menu, type the number corresponding to Backend.
5. On the **Backend Management** menu, enter the option to view and edit an existing backend.
6. Next, you should see a list of the accessible backends on your system. For example, you may see options for the `changelog` and `userRoot` backends. Enter the option to work with the changelog backend.
7. On the **Changelog Backend properties** menu, type the number corresponding to the Enabled property.
8. On the **Enabled Property** menu, type the number to change the `Enabled` property to TRUE.
9. On the **Backend Properties** menu, type `f` to apply the change. If you set up the server in a server group, type `g` to update all of the servers in the group. Otherwise, repeat steps 1-10 on the other servers.
10. Verify that changes made to the data are recorded in the changelog.

Changing the LDAP Changelog Database Location

In cases where disk space issues arise, you can change the on-disk location of the LDAP Change Log database. The changelog backend supports a `db-directory` property that specifies the absolute or relative path (i.e., relative to the local server root) to the filesystem directory that is used to hold the Oracle Berkeley DB Java Edition database files containing the data for this backend.

If you change the changelog database location, you must stop and then restart the Directory Server for the change to take effect. If the changelog backend is already enabled, then the database files must be manually moved or copied to the new location while the server is stopped.

To Change the LDAP Changelog Location Using dsconfig Non-Interactive Mode

Content-Reference to:

1. Use `dsconfig` to change the database location for the LDAP Changelog, which by default is at `<server-root>/db`. The following command sets the LDAP changelog backend to `<server-root>/db2`. Remember to include the LDAP connection parameters (e.g., hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "db-directory:db2" --set "enabled:true"
```

The database files are stored under `<server-root>/db2/changelog`. The files for this backend are stored in a sub-directory named after the `backend-id` property.

2. Stop and restart the server. Since the LDAP changelog backend was previous disabled, there is no need to manually relocate any existing database files.

```
$ bin/stop-server
$ bin/start-server
```

To Reset the LDAP Changelog Location Using dsconfig Non-Interactive Mode

1. If you have changed the LDAP Changelog location, but want to reset it to its original location, use `dsconfig` to reset it. The following command resets the LDAP changelog backend to `<server-root>/db` location. Remember to include the LDAP connection parameters (e.g., hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --reset "db-directory"
```

2. The server attempts to use whatever it finds in the configured location when it starts. If there is nothing there, it will create an empty database. If the LDAP changelog backend at the previous location is enabled at the time, stop the server, manually copy the database files to the new LDAP changelog location, and then restart the server.

Viewing the LDAP Changelog Parameters in the Root DSE

The Root DSE is a special entry that holds operational information about the server. The entry provides information about the LDAP controls, extended operations, and SASL mechanisms available in the server as well as the state of the data within the changelog. For changelog parameters, the attributes of interest include:

- **firstChangeNumber**. Change number for the first (oldest) change record contained in the LDAP changelog.
- **lastChangeNumber**. Change number for the last (most recent) change record contained in the LDAP changelog.
- **lastPurgedChangeNumber**. Change number for the last change that was purged from the LDAP changelog. It can be 0 if no changes have yet been purged.
- **firstReplicaChange**. Information about the first (oldest) change record for a change received from the specified replica. This is a multi-valued attribute and should include a value for each server in the replication topology.
- **lastReplicaChange**. Information about the last (most recent) change record for a change received from the specified replica.

The `firstReplicaChange` and `lastReplicaChange` attributes use the following syntax:

```
serverID:CSN:changeNumber
```

where:

- **serverID**. Specifies the unique identifier for the server updating the change log.
- **CSN**. Specifies the Change Sequence Number, which is the time when the update was made to the given replica.
- **changeNumber**. Specifies the order of the change that is logged to the LDAP changelog.

The `firstReplicaChange` and `lastReplicaChange` attributes can be used to correlate information in the local LDAP Change Log with data in the LDAP Change Log of other servers in the replication topology. The order of the individual changes in the LDAP Change Log can vary between servers based on the order in which they were received from a replica.

To View the LDAP Changelog Parameters

- Use `ldapsearch` to view the Root DSE.

```
$ bin/ldapsearch --baseDN "" --searchScope base "(objectclass=*)" "+"
```

Viewing the LDAP Changelog Using ldapsearch

All records in the changelog are immediate children of the `cn=changelog` entry and are named with the `changeNumber` attribute. You can view changelog entries using `ldapsearch`. Changes are represented in the form documented in the *draft-good-ldap-changelog* specification with the targetDN attribute providing the DN of the updated entry, the `changeType` attribute providing the type of operation (add, delete, modify, or modDN), and the changes attribute providing a base64-encoded representation of the attributes included in the entry (for add operations) or the changes made (for modify operations) in LDIF form. You can view the changes by decoding the encoded value using the `base64` decode utility. The LDAP SDK for Java also provides support for parsing changelog entries.

To View the LDAP Changelog Using ldapsearch

1. Grant access to the `cn=changelog` backend to the `uid=admin` account using access control rules. By default, only the root user has access to this backend.

```
$ bin/ldapmodify
dn: cn=changelog
changetype: modify
add: aci
aci: (targetattr="*||+")
  (version 3.0; acl "Access to the changelog backend for the admin
 account";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

2. Use `ldapsearch` to view the changelog.

```
$ bin/ldapsearch --baseDN cn=changelog --dontWrap "(objectclass=*)"
```

```
dn: cn=changelog
objectClass: top
objectClass: untypedObject
cn: changelog

dn: changeNumber=1,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.0,ou=People,dc=example,dc=com
changeType: modify
changes::
 cmVwbGFjZTogbW9iaWxlCm1vYmlsZTogKzEgMDIwIDE1NCA5Mzk4Ci0KcmVwbGFjZToga
G9tZVBob25lCmhvbWVQaG9uZTogKzEgMjI1IDIxNiA0OTQ5Ci0KcmVwbGFjZTogZ2l2ZW5OYW1lCmdp
dmVuTmFtZTogQWFyb24KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogdGhpcyBpcyB
0aGUgZGVzY3JpcHRpb24gZm9yIEFhcm9uIEF0cC4KLQpyZXBsYWNlOiBtb2RpZmllcnNOYW1lCm1vZG
lmaWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VyLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwotCnJlc
GxhY2U6IGRzLXVwZGF0ZS10aW1lCmRzLXVwZGF0ZS10aW1lOjogQUFBQkhQOHpUR0E9Cgo=
changenumber: 1
dn: changeNumber=2,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: dc=example,dc=com
changeType: modify
changes::
 cmVwbGFjZTogZHMtc3luYy1zdGF0ZQpkcy1zeW5jLXN0YXRlOiAwMDAwMDExQ0ZGMzM0Q
zYwNDA5MzAwMDAwMDAyCgo=
changenumber: 2
```

To View the LDAP Change Sequence Numbers

- The changelog displays the server state information, which is important for failover between servers during synchronization operations. The server state information is exchanged between the servers in the network (LDAP servers and replication servers) as part of the protocol start message. It also helps the client application determine which server is most up-to-date. Make sure that the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

```
$ bin/ldapsearch --baseDN cn=changelog --dontWrap "(objectclass=*)" "+"
```

```
dn: cn=changelog

dn: changeNumber=1,cn=changelog
entry-size-bytes: 182
targetUniqueId: 68147342-1f61-3465-8489-3de58c532130
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184D9E303000000001
replicationCSN: 0000011D27184D9E303000000001
replicaIdentifier: 12336

dn: changeNumber=2,cn=changelog
entry-size-bytes: 263
targetUniqueId: 4e9b7847-edcb-3791-b11b-7505f4a55af4
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184F2E303000000002
replicationCSN: 0000011D27184F2E303000000002
replicaIdentifier: 12336
```

To View LDAP Changelog Monitoring Information

The changelog contains a monitor entry that you can access over LDAP, JConsole, the Administrative Console, or SNMP. Make sure that the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

- Use `ldapsearch` to view the changelog monitor entry.

```
$ bin/ldapsearch --baseDN cn=changelog,cn=monitor "(objectclass=*)"
```

```
dn: cn=changelog,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: extensibleObject
cn: changelog
changelog: cn=changelog
firstchangenumber: 1
lastchangenumber: 8
lastpurgedchangenumber: 0
firstReplicaChange: 16225:0000011D0205237F3F6100000001:5
firstReplicaChange: 16531:0000011CFF334C60409300000002:1
lastReplicaChange: 16225:0000011D02054E8B3F6100000002:7
lastReplicaChange: 16531:0000011CFF334C60409300000002:1
oldest-change-time: 20081015063104Z
...(more data)...
```

Indexing the LDAP Changelog

The Directory Server supports attribute indexing in the Changelog Backend to allow Get Changelog Batch requests to filter results that include only changes involving specific attributes. Normally, the directory server that receives a request must iterate over the whole range of changelog entries and then match entries based on search criteria for inclusion in the batch. The majority of this processing also involves determining whether the changelog entry includes changes to a particular attribute or set of attributes, or not. Using changelog indexing, client applications can dramatically speed up throughput when targeting the specific attributes.

Administrators can configure attribute indexing using the `index-include-attribute` and `index-exclude-attribute` properties on the Changelog Backend. The properties can accept the specific attribute name or special LDAP values "*" to specify all user attributes or "+" to specify all operational attributes.

To determine if the directory server supports this feature, administrators can view the Root DSE for the following entry:

```
supportedFeatures: 1.3.6.1.4.1.30221.2.12.3
```

To Index a Changelog Attribute

1. Use `dsconfig` to set attribute indexing on an attribute in the Changelog Backend.

   The following command enables the Changelog Backend and sets the backend to include all user attributes ("*") for ADD or MODIFY operations using the `changelog-include-attribute` property. The `changelog-deleted-entry-include-attribute` property is set to all attributes ("*") to specify a set of attribute types that should be included in a changelog entry for DELETE operations. Attributes specified in this list will be recorded in the `deletedEntryAttrs` attribute on the changelog entry when an entry is deleted. The attributes `displayName` and `employeeNumber` are indexed using the `index-include-attribute` property.

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --set "enabled:true" \
     --set "changelog-include-attribute:*" \
     --set "changelog-deleted-entry-include-attribute:*" \
     --set "index-include-attribute:displayName" \
     --set "index-include-attribute:employeeNumber"
   ```
2. Add another attribute to index using the `dsconfig` --add option, which adds the attribute to an existing configuration setting.

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --add "index-include-attribute:cn"
   ```

To Exclude Attributes from Indexing

- Use `dsconfig` to set attribute indexing on all user attributes in the Changelog Backend. The following command includes all user attributes except the description and location attributes.

  ```
  $ bin/dsconfig set-backend-prop --backend-name changelog \
    --set "index-include-attribute:*" \
    --set "index-exclude-attribute:description \
    --set "index-exclude-attribute:location
  ```

Tracking Virtual Attribute Changes in the LDAP Changelog

By default, the LDAP Changelog tracks changes to real attributes only. For client applications that require change tracking to include virtual attributes, administrators can enable the `include-virtual-attribute` property, so that real and virtual attributes are tracked within the changelog. Once the `include-virtual-attribute` property is enabled, then properties for virtual attributes that store before/after values, key attributes, and added or deleted entry attributes can be enabled.

To Track Virtual Attribute Changes in the LDAP Changelog

- Use `dsconfig` to enable virtual attribute change tracking in the LDAP Changelog.

  The following command enables the LDAP changelog and sets `include-virtual-attributes` to `add-attributes`, which indicates that virtual attribute be included in the set of attributes listed for an add operation.

The `delete-entry-attributes` option indicates that virtual attributes should be included in the set of deleted entry attributes listed for a delete operation. The `before-and-after-values` option indicates that virtual attributes should be included in the set of before and after values for attributes targeted by the changes. The `key-attribute-values` option indicates that virtual attributes should be included in the set of entry key attribute values.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "enabled:true" \
  --set "include-virtual-attributes:add-attributes" \
  --set "include-virtual-attributes:deleted-entry-attributes" \
  --set "include-virtual-attributes: before-and-after-values" \
  --set "include-virtual-attributes: key-attribute-values"
```

Chapter 16

Managing Access Control

Topics:

The PingDirectory Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and provides examples that illustrate the use of key access control functionality.

Overview of Access Control

The access control model uses access control instructions (ACIs), which are stored in the `aci` operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and affects the entry or any subentries within that branch of the directory information tree (DIT).

Access control instructions specifies four items:

- **Resources**. *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the `cn` or `userPassword` password.
- **Name**. *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."
- **Clients**. *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: `groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."`
- **Rights**. *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant `read` or `write` permission to a `telephoneNumber` attribute.

Key Access Control Features

The PingDirectory Server provides important access control features that provide added security for the Directory Server's entries.

Improved Validation and Security

The Directory Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed `aci` value will be rejected.

As an additional level of security, the Directory Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the `lockdown-mode` privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to an access control instruction not being enforced. When the problem has been corrected, the administrator can use the `leave-lockdown-mode` tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the server (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the PingDirectory Server, global ACIs are defined within the server configuration, in the `global-aci` property of configuration object for the access control handler. They can be viewed and managed using configuration tools like `dsconfig` and the Administrative Console.

The global ACIs available by default in the PingDirectory Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: `namingContexts`, `subschemaSubentry`, `supportedAuthPasswordSchemes`, `supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedLDAPVersion`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`.

- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

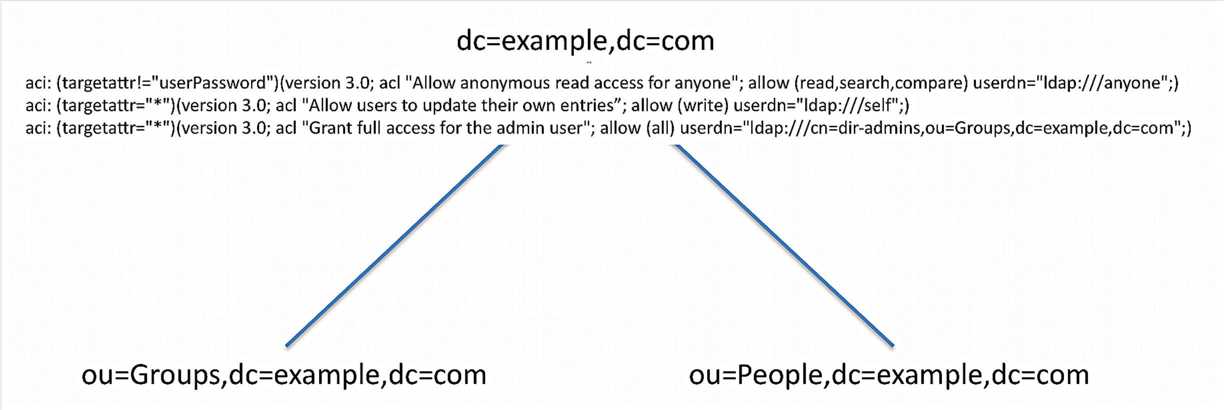
Access Controls for Public or Private Backends

The PingDirectory Server classifies backends as either public or private, depending on their intended purpose. A private backend is one whose content is generated by the Directory Server itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by the server (for example, the LDAP changelog backend). A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The PingDirectory Server access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General Format of the Access Control Rules

Access control instructions (ACIs) are represented as strings that are applied to one or more entries within the Directory Information Tree (DIT). Typically, an ACI is placed on a subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Directory Server iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissable operations. When a client application, such as `ldapsearch`, enters a request, the Directory Server checks that the user who binds with the server has the necessary access rights to the requested search targets. ACIs are cumulatively applied, so that a user who may have an ACI at an entry, may also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.



```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous read access for anyone"; allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")(version 3.0; acl "Allow users to update their own entries”; allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin user"; allow (all) userdn="ldap:///cn=dir-admins,ou=Groups,dc=example,dc=com";)
```

Figure 5: ACI

An access control rule has a basic syntax as follows:

```
aci : (targets) (version 3.0; acl "name"; permissions bind rules;)
```

Table 28: Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and/or attributes to which an access control rule applies. Syntax: *(target keyword = || != expression)*
name	Specifies the name of the ACI.
permissions	Specifies the type of operations to which an access control rule might apply. Syntax: *allow||deny (permission)*
bind rules	Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: *bind rule keyword = ||!= expression;*. The bind rule syntax requires that it be terminated with a ";".

Summary of Access Control Keywords

This section provides an overview of the keywords supported for use in the PingDirectory Server access control implementation.

Targets

A target expression specifies the set of entries and/or attributes to which an access control rule applies. The *keyword* specifies the type of target element. The *expression* specifies the items that is targeted by the access control rule. The operator is either the equal ("=") or not-equal ("!="). Note that the "!=" operator cannot be used with `targattrfilters` and `targetscope` keywords. For specific examples on each target keyword, see the section Working with Targets.

```
(keyword [=||!=]expression)
```

The following keywords are supported for use in the target portion of ACIs:

Table 29: Summary of Access Control Target Keywords

Target Keyword	Description	Wildcards
extop	Specifies the OIDs for any extended operations to which the access control rule should apply.	No
target	Specifies the set of entries, identified using LDAP URLs, to which the access control rule applies.	**Yes**
targattrfilters	Identifies specific attribute values based on filters that may be added to or removed from entries to which the access control rule applies.	**Yes**
targetattr	Specifies the set of attributes to which the access control rule should apply.	**Yes**
targetcontrol	Specifies the OIDs for any request controls to which the access control rule should apply.	No
targetfilter	Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.	**Yes**
targetscope	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI fi there is no target, to which the access control rule should apply.	No

Permissions

Permissions indicate the types of operations to which an access control rule might apply. You can specify if the user or group of users are allowed or not allowed to carry out a specific operation. For example, you would grant read access to a targeted entry or entries using "allow (read)" permission. Or you can specifically deny access to the target

entries and/or attributes using the "deny (read)" permission. You can list out multiple permissions as required in the ACI.

```
allow (permission1 ...,permission2,...permissionN)
```

```
deny (permission1 ...,permission2,...permissionN)
```

The following keywords are supported for use in the permissions portion of ACIs:

Table 30: Summary of Access Control Permissions

Permission	Description
add	Indicates that the access control should apply to add operations.
compare	Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
delete	Indicates that the access control should apply to delete operations.
export	Indicates that the access control should only apply to modify DN operations in which an entry is moved below a different parent by specifying a new superior DN in the modify DN request. The requestor must have the `export` permission for operations against the entry's original DN. The requestor must have the `import` permission for operations against the entry's new superior DN. For modify DN operations that merely alter the RDN of an entry but keeps it below the same parent (i.e., renames the entry), only the `write` permission is required. This is true regardless of whether the entry being renamed is a leaf entry or has subordinate entries.
import	See the description for the `export` permission.
proxy	Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identify to be specified).
read	Indicates that the access control rule should apply to search result entries returned by the server.
search	Indicates that the access control rule should apply to search operations with a non-base scope.
selfwrite	Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
write	Indicates that the access control rule should apply to modify and modify DN operations.
all	An aggregate permission that includes all other permissions except “proxy.” This is equivalent to providing a permission of “add, compare, delete, read, search, selfwrite, write, export, and import.”

Bind Rules

The Bind Rules indicate whether an access control rule should apply to a given requester. The syntax for the target keyword is shown below. The *keyword* specifies the type of target element. The expression specifies the items that is targeted by the access control rule. The operator is either equals ("=") or not-equals ("!="). The semi-colon delimiter symbol (";") is required after the end of the final bind rule.

```
keyword [=||!= ] expression;
```

Multiple bind rules can be combined using boolean operations (AND, OR, NOT) for more access control precision. The standard Boolean rules for evaluation apply: innermost to outer parentheses first, left to right expressions, NOT before AND or OR. For example, an ACI that includes the following bind rule targets all users who are not `uid=admin,dc=example,dc=com` and use simple authentication.

```
(userdn!="ldap:///uid=admin,dc=example,dc=com" and authmethod="simple");
```

The following bind rule targets the `uid=admin,dc=example,dc=com` and authenticates using SASL EXTERNAL or accesses the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or
 ip="127.0.0.1"));
```

The following keywords are supported for use in the bind rule portion of ACIs:

Table 31: Summary of Bind Rule Keywords

Bind Rule Keyword	Description

authmethod	Indicates that the requester's authentication method should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	`authmethod` `=` *`method`*
	where *method* is one of the following representations:
	none
	simple. Indicates that the client is authenticated to the server using a bind DN and password.
	ssl. Indicates that the client is authenticated with an SSL/TLS certificate (e.g., via SASL EXTERNAL), and not just over a secure connection to the server.
	sasl {sasl_mechanism}. Indicates that the client is authenticated to the server using a specified SASL Mechanism.
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) to update their own entries:
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write) (userdn="ldap:///self" and authmethod="ssl");)`

dayofweek	Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. Multiple day of week values may be separated by commas. The keyword's syntax is as follows:
	`dayofweek` `=` *`day1, day2, ...`*
	where *day* is one of the following representations:
	sun
	mon
	tues
	wed
	thu
	fri
	sat
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) on weekdays to update their own entries:
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`

Bind Rule Keyword	Description

	`   allow (write) (dayofweek!="sun,sat" and userdn="ldap:///`
	`self"`
	`   and authmethod="ssl");)`

dns	Indicates that the requester's DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple DNS patterns may be separated by commas. The keyword's syntax is as follows:
	`dns = dns-host-name`
	The following example allows users on hostname `server.example.com` to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (dns="server.example.com" and userdn="ldap:///`
	`self");)`

groupdn	Indicates that the requester's group membership should be taken into account when determining whether the access control rule should apply to any operation. Wildcards are not allowed in this expression.
	`groupdn [ = || != ] "ldap:///groupdn [ || ldap:///`
	`groupdn ] ..."`
	The following example allows users in the managers group to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write)`
	`   (groupdn="ldap:///`
	`cn=managers,ou=groups,dc=example,dc=com");)`

ip	Indicates that the requester's IP address should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple IP address patterns may be separated by commas. The keyword's syntax is as follows:
	`ip [ = || != ] ipAddressList`
	where *ipAddressList* is one of the following representations:
	A specific IPv4 address: 127.0.0.1
	An IPv4 address with wildcards to specify a subnetwork: 127.0.0.*
	An IPv4 address or subnetwork with subnetwork mask: 123.4.5.0+255.255.255.0
	An IPv4 address range using CIDR notation: 123.4.5.0/24
	An IPv6 address as defined by RFC 2373.
	The following example allows users on 10.130.10.2 and localhost to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (ip="10.130.10.2,127.0.0.1" and`
	` userdn="ldap:///self");)`

<table>
<tr><th>Bind Rule Keyword</th><th>Description</th></tr>
<tr><td>timeofday</td><td>
<p>Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword’s syntax is as follows:</p>
<pre>
timeofday [= || != || >= || > || <= || <] time
</pre>
<p>where <em>time</em> is one of the following representations:</p>
<p>4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour)
Wildcards are not allowed in this expression</p>
<p>The following example allows users to update their own entries if the request is received before 12 noon.</p>
<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow users who authenticate before noon
    to update their own entries";
    allow (write) (timeofday<1200 and userdn="ldap:///self"
    and authmethod="simple");)
</pre>
</td></tr>
<tr><td>userattr</td><td>
<p>Indicates that the requester’s relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of <code>USERDN</code> indicates that the target attribute should have a value which matches the DN of the authenticated user. A bindType value of <code>GROUPDN</code> indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A bindType value of <code>LDAPURL</code> indicates that the target attribute should have a value that is an LDAP URL whose criteria matches the entry for the authenticated user. Any value other than <code>USERDN</code>, <code>GROUPDN</code>, or <code>LDAPURL</code> is expected to be present in the target attribute of the authenticated user’s entry. The keyword’s syntax is as follows:</p>
<pre>
userattr = attrName# [bindType || attrValue]
</pre>
<p>where:</p>
<p><em>attrName</em> = name of the attribute for matching
<em>bindType</em> = USERDN, GROUPDN, LDAPURL
<em>attrValue</em> = an attribute value. Note that the <em>attrVALUE</em> of the attribute must match on both the bind entry and the target of the ACI.</p>
<p>The following example allows a manager to change employee's entries. If the bind DN is specified in the <em>manager</em> attribute of the targeted entry, the bind rule is evaluated to TRUE.</p>
<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow a manager to change employee
 entries";
   allow (write) userattr="manager#USERDN";)
</pre>
<p>The following example allows any member of a group to change employee's entries. If the bind DN is a member of the group specified in the <em>allowEditors</em> attribute of the targeted entry, the bind rule is evaluated to TRUE.</p>
<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow allowEditors to change employee
 entries";
   allow (write) userattr="allowEditors#GROUPDN";)
</pre>
</td></tr>
</table>

Bind Rule Keyword	Description

	The following example allows allows a user's manager to edit that user's entry and any entries below the user's entry up to two levels deep. You can specify up to five levels (0, 1, 2, 3, 4) below the targeted entry, with zero (0) indicating the targeted entry.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow managers to change employees entries`
	` two levels below";`
	`    allow (write) userattr="parent[0,1,2].manager#USERDN";)`
	The following example allows any member of the engineering department to update any other member of the engineering department at or below the specified ACI.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow any member of Eng Dept to update any`
	` other member of the`
	`   enginering department at or below the ACI";`
	`   allow (write) userattr="department#ENGINEERING";)`
	The following example allows an entry to be updated by any user whose entry matches the criteria defined in the LDAP URL contained in the `allowedEditorCriteria` attribute of the target entry.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a user that matches the filter to`
	` change entries";`
	`   allow (write) userattr="allowedEditorCriteria#LDAPURL";)`

userdn	Indicates that the user’s DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword’s syntax is as follows:
	`userdn [ = || != ] "ldap:///value [ || "ldap:///value ..."]`
	where *value* is one of the following representations:
	The DN of the target user
	A value of `anyone` to match any client, including unauthenticated clients.
	A value of `all` to match any authenticated client.
	A value of `parent` to match the client authenticated as the user defined in the immediate parent of the target entry.
	A value of `self` to match the client authenticated as the user defined in the target entry.
	If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.
	The following example allows users to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`    allow (write) userdn="ldap:///self";)`

Working with Targets

The following section presents a detailed look and examples of the target ACI keywords: `target`, `targetattr`, `targetfilter`, `targattrfilters`, `targetscope`, `targetcontrol`, and `extop`.

target

The `target` keyword indicates that the ACI should apply to one or more entries at or below the specified distinguished name (DN). The target DN must be equal or subordinate to the DN of the entry in which the ACI is placed. For example, if you place the ACI at the root of `ou=People,dc=example,dc=com`, you can target the DN, `uid=user.1,ou=People,dc=example,dc=com` within your ACI rule. The DN must meet the string representation specification of distinguished names, outlined in RFC 4514, and requires that special characters be properly escaped.

The `target` clause has the following format, where DN is the distinguished name of the entry or branch:

```
(target = ldap:///DN)
```

For example, to target a specific entry, you would use a clause such as the following:

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

Note that, in general, specifying a target DN is not recommended. It is better to have the ACI defined in that entry and omit the `target` element altogether. For example, although you can have `(target="ldap:///uid=john.doe,ou=People,dc=example,dc=com)` in any of the `dc=example,dc=com` or `ou=People` entries, it is better for it to be defined in the `uid=john.doe` entry and not explicitly include the `target` element.

The expression allows for the "not equal" (!=) operator to indicate that all entries within the scope of the given branch that do NOT match the expression be targeted for the ACI. Thus, the following expression targets all entries within the subtree that do not match `uid=john.doe`.

```
(target != ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

The `target` keyword also supports the use of asterisk (*) characters as wildcards to match elements within the distinguished name. The following target expression matches all entries that contains and begins with "john.d, " so that entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=People,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
```

The following target expression matches all entries whose DN begins with "john.d," and matches the `ou` attribute. Entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=asia-branch,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
```

Another example of a complete ACI targets the entries in the `ou=People,dc=example,dc=com` branch and the entries below it, and grants the users the privilege to modify all of their user attributes within their own entries.

```
aci:(target="ldap:///ou=People,dc=example,dc=com")
  (targetattr="*")
  (version 3.0; acl "Allow all the ou=People branch to modify their own
 entries";
 allow (write) userdn="ldap:///self";)
```

targetattr

The `targetattr` keyword targets the attributes for which the access control instruction should apply. There are four general forms that it can take in the PingDirectory Server:

- **(targetattr="*")**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **(targetattr="+")**. Indicates that the access control rule applies to all operational attributes. User attributes will not automatically be included in this set.
- **(targetattr="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies only to the named set of attributes.

- **(targetattr!="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as user attributes and operational attributes. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes, such as when the entry was created, last modified and by whom. Metadata can also include attributes specifying which password policy applies to the user, or overrided default constraints like size limit, time limit, or look-through limit for that user.

The PingDirectory Server distinguishes between these two types of attributes in its access control implementation. The Directory Server does not automatically grant any access at all to operational attributes. For example, the following clause applies only to user attributes and not to operational attributes:

```
(targetattr="*")
```

You can also target multiple attributes in the entry. The following clause targets the common name (cn), surname (sn) and state (st) attribute:

```
(targetattr="cn||sn||st")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

To include all user and all operational attributes, you use both symbols, as follows:

```
(targetattr="*||+")
```

If there is a need to target a specific operational attribute rather than all operational attributes, then it can be specifically included in the values of the `targetattr` clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or if you want to target all user attributes and a specific operational attribute, then you can use them in the `targetattr` clause, as follows:

```
(targetattr="*||ds-rlim-size-limit")
```

The following ACIs are placed on the `dc=example,dc=com` tree and allows any user anonymous read access to all entries except the `userPassword` attribute. The second ACI allows users to update their own contact information. The third ACI allows the `uid=admin` user full access privileges to all user attributes in the `dc=example,dc=com` subtree.

```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous
   read access for anyone"; allow (read,search,compare) userdn="ldap:///
anyone";)
aci: (targetattr="telephonenumber||street||homePhone||l||st")
  (version 3.0; acl "Allow users to update their own contact info";
   allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin
 user";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

An important note must be made when assigning access to user and operational attributes, which can be outlined in an example to show the implications of the Directory Server not distinguishing between these attributes. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of uid and employeeNumber. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a Directory Server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users including administrators granted full rights by access control rules, but excluding users with the bypass-acl privilege, allow them to edit any other attributes in their own entry including those excluded by rules like uid and employeeNumber in the example above, or add, modify, or delete any entries below his or her own entry.

Because the PingDirectory Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the modify-acl privilege.

Another danger in using the (targetattr!="x") pattern is that two ACIs within the same scope could have two different targetattr policies that cancel each other out. For example, if one ACI has (targetattr!="cn|| sn") and a second ACI has (targetattr!="userPassword"), then the net effect is (targetattr="*"), because the first ACI inherently allows userPassword, and the second allows cn and sn.

targetfilter

The targetfilter keyword targets all attributes that match results returned from a filter. The targetfilter clause has the following syntax:

```
(targetfilter = ldap_filter)
```

For example, the following clause targets all entries that contain "ou=engineering" attribue:

```
(targetfilter = "(ou=engineering)")
```

You can only specify a single filter, but that filter can contain multiple elements combined with the OR operator. The following clause targets all entries that contain "ou=engineering," "ou=accounting," and "ou=marketing."

```
(targetfilter = "(|(ou=engineering)(ou=accounting)(ou=marketing)")
```

The following example allows the user, uid=eng-mgr, to modify the departmentNumber, cn, and sn attributes for all entries that match the filter ou=engineering.

```
aci:(targetfilter="(ou=engineering)")
  (targetattr="departmentNumber||cn||sn")
  (version 3.0; acl "example"; allow (write)
   userdn="ldap:///uid=eng-mgr,dc=example,dc=com";)
```

targattrfilters

The targattrfilters keyword targets specific attribute *values* that match a filtered search criteria. This keyword allows you to set up an ACI that grants or denies permissions on an attribute value if that value meets the filter criteria. The targattrfilters keyword applies to individual values of an attribute, not to the whole attribute. The keyword also allows the use of wildcards in the filters.

The keyword clause has the following formats:

```
(target = "add=attr1:Filter1 && attr2:Filter2... && attrn:FilterN,
del=attr1:Filter1 && attr2:Filter2 ... && attrN:FilterN" )
```

where

add represents the operation of adding an attribute value to the entry

del represents the operation of removing an attribute value from the entry

attr1, attr2... attrN represents the targeted attributes

filter1, filter2 ... filterN represents filters that identify matching attribute values

The following conditions determine when the attribute must satisfy the filter:

- When adding or deleting an entry containing an attribute targeted a `targattrfilters` element, each value of that attribute must satisfy the corresponding filter.
- When modifying an entry, if the operation adds one or more values for an attribute targeted by a `targattrfilters` element, each value must satisfy the corresponding filter. If the operation deletes one or more values for a targeted attribute, each value must satisfy the corresponding filter.
- When replacing the set of values for an attribute targeted by a `targattrfilters` element, each value removed must satisfy the delete filters, and each value added must satisfy the add filters.

The following example allows any user who is part of the `cn=directory server admins` group to add the `soft-delete-read` privilege.

```
aci:(targattrfilter="add=ds-privilege-name:(ds-privilege-name=soft-delete-
read)")
  (version 3.0; acl "Allow members of the directory server admins group to
 grant the
   soft-delete-read privilege"; allow (write)
   groupdn="ldap:///cn=directory server admins,ou=group,dc=example,dc=com";)
```

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is base to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
    (targetattr="supportedControl||supportedExtension||
     supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
    (version 3.0; acl "Allow users to view Root DSE Operational Attributes";
     allow (read,search,compare) userdn="ldap:///anyone")
```

targetcontrol

The `targetcontrol` keyword is used to indicate whether a given request control can be used by those users targeted in the ACI. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying control OIDs.

The following ACI example shows the controls required to allow an administrator to use and manage the Soft-Delete feature. The Soft Delete Request Control allows the user to soft-delete an entry, so that it could be undeleted at a later time. The Hard Delete Request Control allows the user to permanently remove an entry or soft-deleted entry. The Undelete Request Control allows the user to undelete a currently soft-deleted entry. The Soft-Deleted Entry Access Request Control allows the user to search for any soft-deleted entries in the server.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.22||
   1.3.6.1.4.1.30221.2.5.23||1.3.6.1.4.1.30221.2.5.24")
  (version 3.0; acl "Allow admins to use the Soft Delete Request Control,
   Hard Delete Request Control,Undelete Request Control, and
   Soft-deleted entry access request control";
   allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

extOp

The `extop` keyword can be used to indicate whether a given extended request operation can be used. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying extended request OIDs.

The following ACI allows the `uid=user-mgr` to use the Password Modify Request (i.e., OID=1.3.6.1.4.1.4203.1.11.1) and the StartTLS (i.e., OID=1.3.6.1.4.1.1466.20037) extended request OIDs.

```
aci:(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")
  (version 3.0; acl "Allows the mgr to use the Password Modify Request and
 StartTLS;
   allow(read) userdn="ldap:///uid=user-mgr,ou=people,dc=example,dc=com";)
```

Examples of Common Access Control Rules

This section provides a set of examples that demonstrate access controls that are commonly used in your environment. Note that to be able to alter access control definitions in the server, a user must have the `modify-acl` privilege as discussed later in this chapter.

Administrator Access

The following ACI can be used to grant any member of the "`cn=admins,ou=groups,dc=example,dc=com`" group to add, modify and delete entries, reset passwords and read operational attributes such as `isMemberOf` and password policy state:

```
aci: (targetattr="+")(version 3.0; acl "Administrators can read, search or
 compare operational attributes";
allow (read,search,compare) groupdn="ldap:///
cn=admins,ou=groups,dc=example,dc=com";)
aci: (targetattr="*")(version 3.0; acl "Administrators can add, modify and
 delete entries";
allow (all) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

Anonymous and Authenticated Access

The following ACI allow anonymous read, search and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. The authenticated user will inherit the privileges of the anonymous ACI. In addition, the authenticated user can change `userPassword`:

```
aci: (targetattr="objectclass || uid || cn || mail || sn || givenName")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Anyone can access names and email addresses of entries
 representing people";
allow (read,search,compare) userdn="ldap:///anyone";)
```

```
aci: (targetattr="departmentNumber || manager || isMemberOf")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Authenticated users can access these fields for entries
 representing people";
allow (read,search,compare) userdn="ldap:///all";)
aci: (targetattr="userPassword")(version 3.0; acl "Authenticated users can
 change password";
allow (write) userdn="ldap:///all";)
```

If no unauthenticated access should be allowed to the Directory Server, the preferred method for preventing unauthenticated, or anonymous access is to set the Global Configuration property `reject-unauthenticated-requests` to true.

Delegated Access to a Manager

The following ACI can be used to allow an employee's manager to edit the value of the employee's `telephoneNumber` attribute. This ACI uses the `userattr` keyword with a bind type of `USERDN`, which indicates that the target entry's manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
(version 3.0; acl "A manager can update telephone numbers of her direct
 reports";
allow (read,search,compare,write) userattr="manager#USERDN";)
```

Proxy Authorization

The following ACIs can be used to allow the application "`cn=OnBehalf,ou=applications,dc=example,dc=com`" to use the proxied authorization v2 control to request that operations be performed using an alternate authorization identity. The application user is also required to have the `proxied-auth` privilege as discussed later in this chapter:

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com";)
```

Validating ACIs Before Migrating Data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Sun/Oracle encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the PingDirectory Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a PingDirectory Server.

To validate an access control instruction, the PingDirectory Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values, and all other attributes will be ignored.

To Validate ACIs from a File

Content-Reference to:

The `validate-acis` tool can process data contained in an LDIF file. It will ignore all attributes except `aci`, and will ignore all entries that do not contain the `aci` attribute, so any existing LDIF file that contains access control rules may be used.

1. Run the `bin/validate-acis` tool (UNIX or Linux systems) or `bat\validate-acis` (Win dows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

   ```
   $ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-
   acis.ldif
   ```

   ```
   # Processing complete # Total entries examined: 1
   # Entries found with ACIs: 1
   # Total ACI values found: 3
   # Malformed ACI values found: 0
   # Other processing errors encountered: 0
   ```
2. Review the results by opening the output file. For example, the `validated-acis.ldif` file that was generated in the previous step reads as follows:

   ```
   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr!="userPassword")
     (version 3.0; acl "Allow anonymous read access for anyone";
       allow (read,search,compare) userdn="ldap:///anyone";)

   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr="*")
     (version 3.0; acl "Allow users to update their own entries";
       allow (write) userdn="ldap:///self";)

   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr="*")
     (version 3.0; acl "Grant full access for the admin user";
       allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
   ```
3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed ACI syntax is encountered.

   ```
   $ bin/validate-acis --ldifFIle test-acis.ldif --outputFile validated-
   acis.ldif \
     --onlyReportErrors
   ```

   ```
   # Processing complete
   # Total entries examined: 1
   # Entries found with ACIs: 1
   # Total ACI values found: 3
   # Malformed ACI values found: 0
   # Other processing errors encountered: 0
   ```

   The output file shows the following message:

   ```
   # The following access control rule is malformed or contains an unsupported
   # syntax: The provided string '(targetattr!=userPassword)(version 3.0; acl
   # "Allow anonymous read access for anyone"; allow (read,search,compare)
   # userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
   # Instruction (ACI) because it failed general ACI syntax evaluation
   dn: dc=example,dc=com
   aci: (targetattr!=userPassword)
     (version 3.0; acl "Allow anonymous read access for anyone";
       allow (read,search,compare) userdn="ldap:///anyone";)

   # The following access control rule is valid
   ```

```
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Validate ACIs in Another Directory Server

The `validate-acis` tool also provides the ability to examine ACIs in data that exists in another Directory Server that you are planning to migrate to the PingDirectory Server. The tool helps to determine whether the Ping Identity Server accepts those ACIs.

- To use it in this manner, provide arguments that specify the address and port of the target Directory Server, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

Migrating ACIs from Sun/Oracle to PingDirectory Server

This section describes the most important differences in access control evaluation between Sun/Oracle and the PingDirectory Server.

Support for Macro ACIs

Sun/Oracle provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the PingDirectory Server. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN Bind Rule

Sun/Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The PingDirectory Server does not support DSEE roles and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting Operational Attributes

The Sun/Oracle access control model does not differentiate between user attributes and operational attributes. With Sun/Oracle, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible for several significant security holes in which users are unintentionally given access to operational attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the PingDirectory Server, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except

userPassword, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like targetattr="creatorsName||modifiersName". All operational attributes can be targeted using the "+" character. So, targetattr="+" targets all operational attributes but no user attributes and targetattr="*||+" targets all user and operational attributes.

Specification of Global ACIs

Both DSEE and PingDirectory Server support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the aci attribute. In the PingDirectory Server, global ACIs are managed with dsconfig referenced in the global-aci property of the Access Control Handler.

Defining ACIs for Non-User Content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the PingDirectory Server, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting Access to Controls and Extended Operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as "oid={oid},cn=features,cn=config" where {oid} is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control: "oid=2.16.840.1.113730.3.4.3,cn=features,cn=config".

In the PingDirectory Server, the "targetcontrol" keyword can be used to define ACIs that grant or deny access to controls. The "extop" keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for Malformed ACI Values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The PingDirectory Server is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the lockdown-mode privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.

Note: Consider running the import-ldif tool with the --rejectFile option so that you can review any rejected ACIs.

About the Privilege Subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you cannot restrict the capabilities assigned to that root user.

The PingDirectory Server offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server

configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying Unsupported ACIs

The PingDirectory Server provides a `validate-acis` tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the PingDirectory Server instance. When migrating data from a DSEE deployment into a PingDirectory Server instance, the `validate-acis` tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the PingDirectory Server.

For more information about using this tool, see Validating ACIs Before Migrating Data.

Working with Privileges

In addition to the access control implementation, the PingDirectory Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the PingDirectory Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available Privileges

The following privileges are defined in the PingDirectory Server.

Table 32: Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the `audit-data-security` tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the `bypass-acl` privilege but without the `config-read` privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is not evaluated for bind

Privilege	Description
	operations so that password policy evaluation will still occur when binding as a user with this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the `config-read` privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the Directory Server. The user is also required to have the `jmx-read` privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Server via the Java Management Extensions (JMX). The user is also required to have the `jmx-read` privilege. Note that currently all of the information exposed by the server over JMX is read-only.
ldif-export	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group.
ldif-import	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the `ldif-export` privilege.
lockdown-mode	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.
modify-acl	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.
password-reset	This privilege is required for one user to be allowed to change another user’s password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.
privilege-change	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the `ds-privilege-name` operational attribute.

Privilege	Description
proxied-auth	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
server-restart	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
server-shutdown	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
soft-delete-read	This privilege is required for a user to access a soft-deleted-entry.
stream-values	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
unindexed-search	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
update-schema	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges Automatically Granted to Root Users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the `ds-cfg-inherit-default-root-privileges` attribute with a value of FALSE in the entries for those root users.
- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the `default-root-privilege-name` property of the Root DN configuration object. By default, this set of privileges includes:

audit-data-security
backend-backup
backend-restore
bypass-acl
config-read
config-write
disconnect-client
ldif-export
lockdown-mode
manage-topology
metrics-read
modify-acl
password-reset

permit-get-password-policy-state-issues
privilege-change
server-restart
server-shutdown
soft-delete-read
stream-values
unindexed-search
update-schema

The privileges not granted to root users by default includes:

bypass-pw-policy
bypass-read-acl
jmx-read
jmx-write
jmx-notify
permit-externally-processed-authentication
permit-proxied-mschapv2-details
proxied-auth

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the `config-read`, `config-write`, and `privilege-change` privileges, as well as either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning Additional Privileges for Administrators

To allow access to the Tasks backend, set up a global ACI that allows access to members of an Administrators group as follows:

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*||+")
        (version 5.0; acl "Access to the tasks backend for administrators";
         allow (all) groupdn="ldap:///
         cn=admins,ou=groups,dc=example,dc=com";)'
```

Assigning Privileges to Normal Users and Individual Root Users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling Privileges

Although the privilege subsystem in the PingDirectory Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Working with Proxied Authorization

The Directory Server supports the Proxied Authorization Control (RFC 4370) to allow an authorized LDAP client to authenticate to the server as another user. Typically, LDAP servers are deployed as backend authentication systems that store user credentials and authorization privileges necessary to carry out an operation. Single sign-on (SSO) systems can retrieve user credentials from the Directory Server and then issue permissions that allow the LDAP client to request operations under the identity as another user. The use of the proxied authorization control provides a means for client applications to securely process requests without the need to bind or re-authenticate to the server for each and every operation.

The Directory Server supports the proxied authorization V1 and V2 request controls. The proxied authorization V1 request control is based on early versions of the draft-weltman-ldapv3-proxy Internet draft and is available primarily for legacy systems. It is recommended that deployments use the proxied authorization V2 request control based on RFC 4370.

The proxied authorization V2 control is used to request that the associated operation be performed as if it has been requested by some other user. This control may be used in conjunction with add, delete, compare, extended, modify, modify DN, and search requests. In that case, the associated operation will be processed under the authority of the specified authorization identity rather than the identity associated with the client connection (i.e., the user as whom that connection is bound). The target authorization identity for this control is specified as an "authzid" value, which should be either "dn:" followed by the distinguished name of the target user, or "u:" followed by the username.

Note that because of the inherent security risks associated with the use of the proxied authorization control, most directory servers that support its use enforce strict restrictions on the users that are allowed to request this control. If a user attempts to use the proxied authorization V2 request control and does not have sufficient permission to do so, then the server will return a failure response with the `AUTHORIZATION_DENIED` result code.

Configuring Proxied Authorization

Configuring proxied authorization requires a combination of access control instructions and the `proxied-auth` privilege to the entry that will perform operations as another user.

Note: You cannot use the `cn=Directory Manager` root DN as a proxying DN.

To Configure Proxied Authorization

1. Open a text editor and create a user entry, such as `uid=clientApp`, which is the user entry that will request operations as another user, `uid=admin,dc=example,dc=com`. The client application entry also requires the `proxied-auth` privilege to allow it to run proxied authorization requests. Save the file as `add-user.ldif`.

```
dn: ou=Applications,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
objectClass: extensibleObject
ou: Admins
ou: Applications

dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
givenName: Client
uid: clientApp
cn: Client App
sn: App
userPassword: password
ds-privilege-name: proxied-auth
```

2. Add the file using `ldapmodify`.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

3. The client application targets a specific subtree in the Directory Information Tree (DIT) for its operations. For example, some client may need access to an accounts subtree to retrieve customer information. Another client may need access to another subtree, such as a subscriber subtree. In this example, we want the client application to target the `ou=People,dc=example,dc=com` subtree. To allow the target, open a text editor and create an LDIF file to assign an ACI to that branch so that the client app user can access it as a proxy auth user. Note that the ACI should be on a single line of text. The example shows the ACI over multiple lines for readability. Add the file using the `ldapmodify`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci:  (version 3.0; acl "People Proxy Access"; allow(proxy)
  userdn="ldap:///uid=clientApp,ou=Applications,dc=example,dc=com";)
```

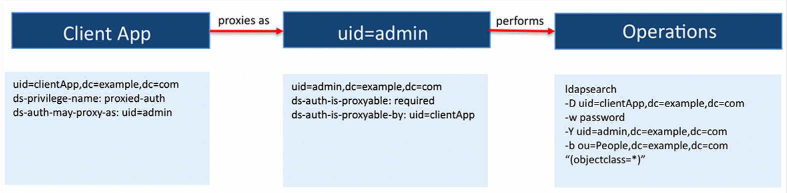
4. Run a search to test the configuration using the bind DN `uid=clientApp` and the `proxyAs` option, which requires that you prefix "dn:" to the proxying entry or "u:" to the username. The `uid=clientApp` binds to the server and proxies as `uid=admin` to access the `ou=People,dc=example,dc=com` subtree.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

Restricting Proxy Users

The Directory Server provides a set of operational attributes that restricts the proxied authorization capabilities of a client application and its proxyable target entry. When present in an entry, the Directory Server evaluates each operational attribute together to form a whitelist of potential users that can be proxied. If none of those attributes is present, then the user may potentially proxy as anyone.

The Directory Server supports a two-tier provision system that, when configured, can restrict specific users for proxied authorization. The first tier is a set of `ds-auth-may-proxy-as-*` operational attributes on the client entry that will bind to the server and carry out operations under the identity of another user. The second tier is a set of `ds-auth-is-proxyable-*` operational attributes on the user entry that defines whether access is allowed, prohibited, or required by means of proxied authorization. If allowed or required, the attributes define which client entries can proxy as the user.



uid=clientApp,dc=example,dc=com
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: uid=admin

uid=admin,dc=example,dc=com
ds-auth-is-proxyable: required
ds-auth-is-proxyable-by: uid=clientApp

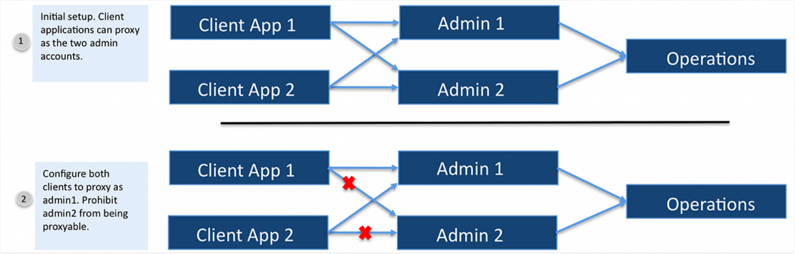
```
ldapsearch
-D uid=clientApp,dc=example,dc=com
-w password
-Y uid=admin,dc=example,dc=com
-b ou=People,dc=example,dc=com
“(objectclass=*)”
```

Figure 6: Proxying Operational Attributes

For example, if a client application, such as `uid=clientApp`, is requesting to search the `ou=People,dc=example,dc=com` branch as the user `uid=admin`, the command would look like this:

```
ldapsearch --bindDN uid=clientApp,dc=example,dc=com \
--bindPassword password \
--proxyAs uid=admin,dc=example,dc=com \
--baseDN ou=People,dc=example,dc=com \
"(object-class=*)
```

At bind, the Directory Server evaluates the list of users in the `uid=clientApp` entry based on the presence of any `ds-auth-may-proxy-as-*` attributes. In the figure below, the `uid=clientApp` entry has a `ds-auth-may-proxy-as` attribute with a value, `uid=admin`, which means that the client app user may proxy only as the `uid=admin` account. Next, the server confirms that `uid=admin` is in the list of proxyable users and then evaluates the `ds-auth-is-proxyable-*` attributes present in the `uid=admin` entry. These attributes determine the list of restricted users that either are allowed, prohibited, or required to proxy as the `uid=admin` entry. In this case, the `uid=admin` entry has the `ds-auth-is-proxyable` attribute with a value of "required", which indicates that the entry can only be accessed by means of proxied authorization. The `uid=admin` entry also has the `ds-auth-is-proxyable-by` attribute with a value of `uid=clientApp`, which indicates it can only be requested by the `uid=clientApp` entry. Once both sets of attributes have been confirmed, the `uid=clientApp` can bind to the server as the authenticated user. From this point, the Directory Server performs ACI evaluation on the branch to determine if the requested user has access rights to the branch. If the branch is accessible by the `uid=clientApp` entry, and then the search request is processed.



Initial setup. Client applications can proxy as the two admin accounts.

Configure both clients to proxy as admin1. Prohibit admin2 from being proxyable.

Figure 7: Proxying Operational Attributes Examples

About the ds-auth-may-proxy-as-* Operational Attributes

The Directory Server first evaluates the list of potential users that can be proxied for the authenticated user based on the presence of the `ds-auth-may-*` operational attributes in the entry. These operational attributes are multi-valued and are evaluated together if all are present in an entry:

- **ds-auth-may-proxy-as**. Specifies the user DNs that the associated user is allowed to proxy as. For instance, based on the previous example, you could specify in the `uid=clientApp` entry that it can proxy operations as `uid=admin` and `uid=agent1`.

  ```
  dn: uid=clientApp,ou=Applications,dc=example,dc=com
  ```

```
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: uid=admin,dc=example,dc=com
ds-auth-may-proxy-as: uid=agent1,ou=admins,dc=example,dc=com
```

- **ds-auth-may-proxy-as-group**. Specifies the group DNs and its group members that the associated user is allowed to proxy as. For instance, you could specify that the potential users that the `uid=clientApp` entry can proxy as are those members who are present in the group `cn=Agents,ou=Groups,dc=example,dc=com`. This attribute is multi-valued, so that more than one group can be specified. Nested static and dynamic groups are also supported.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as-group: cn=Agents,ou=Groups,dc=example,dc=com
```

- **ds-auth-may-proxy-as-url**. Specifies the DNs that are returned based on the criteria defined in an LDAP URL that the associated user is allowed to proxy as. For instance, the attribute specifies that the client can proxy as those entries that match the criteria in the LDAP URL. This attribute is multi-valued, so that more than one LDAP URL can be specified.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as-url: ldap:///ou=People,dc=example,dc=com??sub?
(l=austin)
```

About the ds-auth-is-proxyable-* Operational Attributes

After the Directory Server has evaluated the list of users that the authenticated user can proxy as, the server checks to see if the requested authorized user is in the list. If the requested authorized user is present in the list, then the server continues processing the proxable attributes in the entry. If the requested authorized user is not present in the list, the bind will fail.

The operational attributes on the proxying entry are as follows:

- **ds-auth-is-proxyable**. Specifies whether the entry is proxyable or not. Possible values are: "allowed" (operation may be proxied as this user), "prohibited" (operations may not be proxied as this user), "required" (indicates that the account will not be allowed to authenticate directly but may only be accessed by some form of proxied authorization).
- **ds-auth-is-proxyable-as**. Specifies any users allowed to use this entry as a target of proxied authorization.
- **ds-auth-is-proxyable-as-group**. Specifies any groups allowed to use this entry as a target of proxied authorization. Nested static and dynamic groups are also supported.
- **ds-auth-is-proxyable-as-url**. Specifies the LDAP URLs that are used to determine any users that are allowed to use this entry as a target of proxied authorization.

Restricting Proxied Authorization for Specific Users

To illustrate how the proxied authorization operational attributes work, it is best to set up a simple example where two LDAP clients, `uid=clientApp1` and `uid=clientApp2` can freely proxy two administrator accounts, `uid=admin1` and `uid=admin2`. We will add the `ds-auth-may-proxy-as-*` and the `ds-auth-is-proxyable-*` attributes to these entries to restrict how each account can use proxied authorization. For example, the two client applications will continue to proxy the `uid=admin1` account but the `uid=admin2` account will no longer be able to be used as a proxied entry.

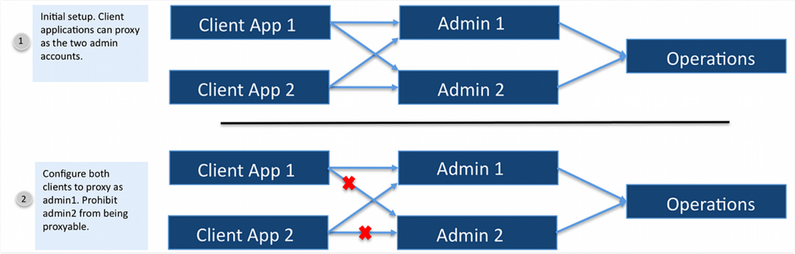

1
Initial setup. Client applications can proxy as the two admin accounts.
Client App 1
Client App 2
Admin 1
Admin 2
Operations
2
Configure both clients to proxy as admin1. Prohibit admin2 from being proxyable.
Client App 1
Client App 2
Admin 1
Admin 2
Operations

Figure 8: Proxy Users Example Scenario

To Restrict Proxied Authorization for Specific Users

1. For this example, set up two user entries, `uid=clientApp1` and `uid=clientApp2`, which will be proxying the `uid=admin1` and `uid=admin2` accounts to access the `ou=People,dc=example,dc=com` subtree. Both entries have the `proxied-auth` privilege assigned to it. Open a text editor and create an LDIF file. Add the file using the `ldapmodify` tool. Note that "..." indicates that other attributes present in the entry are not included in the example for readability purposes.

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth

dn: uid=clientApp2,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
```

2. Next, assign the ACI for each client application to the subtree, `ou=People,dc=example,dc=com`. Note that the ACIs should be on one line of text. The example displays the ACIs over multiple lines for readability.

```
dn: ou=People,dc=example,dc=com
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
       userdn="ldap:///uid=clientApp1,ou=Applications,dc=example,dc=com";)
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
       userdn="ldap:///uid=clientApp2,ou=Applications,dc=example,dc=com";)
```

3. Run a search for each entry. In this example, assume that there are two admin accounts: `admin1` and `admin2` that have full access rights to user attributes. You should be able to proxy as the `uid=admin1` and `uid=admin2` entries to access the subtree for both clients.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin1,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"

$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

4. Next, limit the proxied authorization capabilities for each client application. Update the `uid=clientApp1` entry to add the `ds-auth-may-proxy-as` attribute. In this example, the `ds-auth-may-proxy-as` attribute specifies that `uid=clientApp1` can proxy as the `uid=admin1` entry. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`. Note that `ds-auth-may-proxy-as` is multi-valued:

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
```

```
changetype: modify
add: ds-auth-may-proxy-as
ds-auth-may-proxy-as: uid=admin1,dc=example,dc=com
```

5. Repeat the previous step for the `uid=clientApp2` entry, except specify the `ds-auth-may-proxy-as-url`. The client entry may proxy as any DN that matches the LDAP URL.

```
dn: uid=clientApp2,ou=Applications,dc=example,dc=com
changetype: modify
add: ds-auth-may-proxy-as-url
ds-auth-may-proxy-as-url: ldap:///dc=example,dc=com??sub?(uid=admin*)
```

6. Next, we want to create a group of client applications that has `uid=clientApp1` and `uid=clientApp2` as its `uniquemembers` to illustrate the use of the `ds-auth-proxyable-by-group` attribute. In this example, set up a static group using the `groupOfUniqueNames` object class.

```
dn: ou=Groups,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: groups

dn: cn=Client Applications,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: Client Applications
ou: groups
uniquemember: uid=clientApp1,ou=Applications,dc=example,dc=com
uniquemember: uid=clientApp2,ou=Applications,dc=example,dc=com
```

7. Update the `uid=admin1` entry to provide the DN that it may be proxied as. Add the `ds-auth-is-proxyable` and the `ds-auth-is-proxyable-by` attributes. For instance, we make the `uid=admin1` a required proxyable entry, which means that it can only be accessed by some form of proxied authorization. Then, specify each DN that can proxy as `uid=admin1` using the `ds-auth-is-proxyable-by`. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`. Note that the example includes all three types of `ds-auth-is-proxable-by-*` attributes as an illustration, but, in an actual deployment, only one type of attribute is necessary if they all target the same entries.

```
dn: uid=admin1,dc=example,dc=com
changetype: modify
add: ds-auth-is-proxyable
ds-auth-is-proxyable: required
-
add: ds-auth-is-proxyable-by
ds-auth-is-proxyable-by: ou=clientApp1,ou=Applications,dc=example,dc=com
ds-auth-is-proxyable-by: ou=clientApp2,ou=Applications,dc=example,dc=com
-
add: ds-auth-is-proxyable-by-group
ds-auth-is-proxyable-by-group: cn=Client
 Applications,ou=Groups,dc=example,dc=com
-
add: ds-auth-is-proxyable-by-url
ds-auth-is-proxyable-by-url: ldap:///ou=Applications,dc=example,dc=com??sub?
(uid=clientApp*)
```

8. Next, prohibit proxying for the `uid=admin2` entry by setting the `ds-auth-is-proxyable` to prohibited. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`.

```
dn: uid=admin2,dc=example,dc=com
changetype: modify
add: ds-auth-is-proxyable
ds-auth-is-proxyable: prohibited
```

9. Run a search using the proxied account. For example, run a search first with `uid=clientApp1` or `uid=clientApp2` that proxies as `uid=admin1` to return a successful operation. However, if you run a search

for uid=clientApp1 that proxies as uid=admin2, as seen below, you will see an "authorization denied" message due to uid=admin2 not matching the list of potential entries that can be proxied. The ds-auth-may-proxy-as-* attributes specify that the client can only proxy as uid=admin1:

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

```
One of the operational attributes (ds-auth-may-proxy-as,
ds-auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry
'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow
that user to be proxied as user 'uid=admin2,dc=example,dc=com'

Result Code:  123 (Authorization Denied)

Diagnostic Message:  One of the operational attributes (ds-auth-may-proxy-
as,
ds-auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry
'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow that
user to be proxied as user 'uid=admin2,dc=example,dc=com'
```

10. Run another search using uid=clientApp2, which attempts to proxy as uid=admin2. You will see an "authorization denied" message due to the presence of the ds-auth-is-proxyable:prohibited operational attribute, which states that uid=admin2 is not available for proxied authorization.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

```
The 'ds-auth-is-proxyable' operational attribute
in user entry 'uid=admin2,dc=example,dc=com' indicates that
user may not be accessed via proxied authorization

Result Code:  123 (Authorization Denied)

Diagnostic Message: The 'ds-auth-is-proxyable' operational
attribute in user entry 'uid=admin2,dc=example,dc=com' indicates
that user may not be accessed via proxied authorization
```

Working with Parameterized ACIs

The Directory Server supports the use of parameterized ACIs to control access to subtrees with homogenous administrative group or user patterns, which can be used in multi-tenant deployments, such as PingOne for Customers. A single parameterized ACI can take the place of specifying identical ACIs on each tenant's subtree. For example, the following parameterized ACI:

```
 (target="ldap:///o=($1),dc=example,dc=com")(version 3.0; acl \
"Subtree Admin Group members may search for and read entries in their
 subtree."; allow \
(search, read) groupdn="ldap:///cn=Subtree Admin
 Group,ou=groups,o=($1),dc=example,dc=com";)
```

Enables:

- Members of a group with DN "cn=Subtree Admin Group,ou=groups,o=Customers, dc=example,dc=com" to search for and read entries in the "o=Customers, dc=example,dc=com" subtree.
- Members of a group with DN "cn=Subtree Admin Group,ou=groups,o=Partners, dc=example,dc=com" to search for and read entries in the "o=Partners, dc=example,dc=com" subtree

The same access is granted for any substitution value for the ($1) parameter variable. If an operation tried to read the uid=user.1,o=acme,dc=example,dc=com entry, this ACI would be considered. This ACI would allow a read action, if the operation's user is a member of the cn=Subtree Admin Group,ou=groups,o=acme,dc=example,dc=com group.

Attribute values from the target DN can be replaced with different variables ($#) and then reference those variables in the group DN or user DN. The string representation of a parameter variable is constructed as follows:

- an open parenthesis
- a dollar sign
- a positive integer
- a closing parenthesis

In another example:

```
"population=($2),ou=Populations,environment=($1),ou=Environments,o=Acme"
```

The ($2) variable is the population ID in the DN of the target entry, and ($1) is the environment ID in the DN of the target entry. Those values from the target entry's DN are then substituted into the group DN or user DN value.

Parameter variables present in a parameterized ACI's target will be associated with the actual values from the resource DN. Each actual value will be substituted for its respective parameter variable in the ACI's target, and group bind rule DNs when performing access control on the resource entry. Parameter variables can be used in multiple RDNs in a parameterized target. A given RDN may have at most one parameter variable as its attribute value, and a given parameter variable may appear only once in the parameterized target.

The following values are examples of valid parameterized target DNs:

- ou=($1),dc=example,dc=com
- population=($2),ou=Populations,environment=($1),ou=Environments,o=Acme
- o=($1) (for a global ACI)

An ACI on an entry can only apply to that entry's subtree. If an ACI with a parameterized target is stored on an entry, that entry's DN must appear in a non-parameterized form as the rightmost RDNs of the parameterized target's DN. For example, if an ACI with a parameterized target were stored on the dc=example,dc=com entry, that parameterized target must end in dc=example,dc=com in a non-parameterized form. Global ACIs do not have this restriction. Each global ACI can have parameter variables in any or all of its parameterized target's RDNs. Additional restrictions for parameterized targets include:

- They may not be pattern ACIs. That is, they may not contain wildcards ('*').
- RDNs that are parameterized must be single-valued. For example, a given parameterized RDN may not consist of two or more type-value pairs joined by '+'.

Chapter 17

Managing the Schema

Topics:

This chapter presents a basic summary of the supported schema components on the PingDirectory Server and procedures to extend the schema with new element definitions. The chapter presents the following topics:

About the Schema

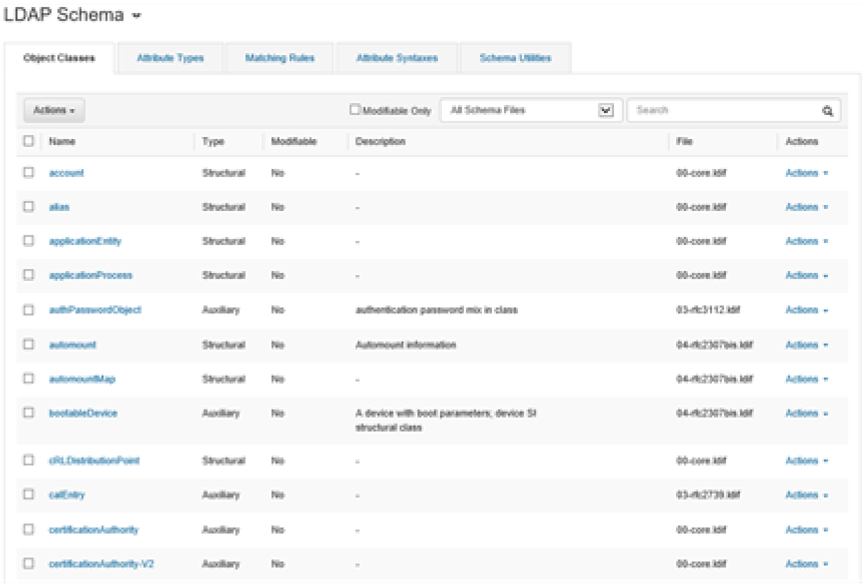
A schema is the set of directory server rules that define the structures, contents, and constraints of a Directory Information Tree (DIT). The schema guarantees that any new data entries or modifications meet and conform to these predetermined set of definitions. It also reduces redundant data definitions and provides a uniform method for clients or applications to access its Directory Server objects.

The PingDirectory Server ships with a default set of read-only schema files that define the core properties for the Directory Server. The Administrative Console provides a Schema Editor that administrators can use to view existing schema definitions and add new custom schema elements to their DIT. Any attempt to alter a schema element defined in a read-only file or add a new schema element to a read-only file will result in an "Unwilling to Perform" result.

About the Schema Editor

The Administrative Console Manangement Console provides a user-friendly graphical editor with tabs to manage any existing schema component related to the DIT: object classes, attributes, matching rules, attribute syntaxes, and schema utilities. The **Object Classes** and **Attribute Types** tabs enable viewing existing definitions as well as adding, modifying, or removing custom schema elements.

The **Matching Rules** and **Attribute Syntaxes** tabs are read-only and provide a comprehensive listing of all of the elements necessary to define new schema elements. The **Schema Utilities** tab provides a schema validator that allows you to load a schema file or perform a cut-and-paste operation on the schema definition to verify that it meets the proper schema and ASN.1 formatting rules. The **Utilities** tab also supports schema file imports by first checking for proper syntax compliance and generating any error message if the definitions do not meet specification.



□	Name	Type	Modifiable	Description	File	Actions
□	account	Structural	No	-	00-core.ldif	Actions ▾
□	alias	Structural	No	-	00-core.ldif	Actions ▾
□	applicationEntity	Structural	No	-	00-core.ldif	Actions ▾
□	applicationProcess	Structural	No	-	00-core.ldif	Actions ▾
□	authPasswordObject	Auxiliary	No	authentication password mix in class	03-rfc3112.ldif	Actions ▾
□	automount	Structural	No	Automount information	04-rfc2307bis.ldif	Actions ▾
□	automountMap	Structural	No	-	04-rfc2307bis.ldif	Actions ▾
□	bootableDevice	Auxiliary	No	A device with boot parameters; device SI structural class	04-rfc2307bis.ldif	Actions ▾
□	cRLDistributionPoint	Structural	No	-	00-core.ldif	Actions ▾
□	calEntry	Auxiliary	No	-	03-rfc2739.ldif	Actions ▾
□	certificationAuthority	Auxiliary	No	-	00-core.ldif	Actions ▾
□	certificationAuthority-V2	Auxiliary	No	-	00-core.ldif	Actions ▾

Figure 9: Example Schema Editor Screen

The Schema Editor provides two views for each definition: **Properties View** and **LDIF View**. The **Properties View** breaks down the schema definition by its properties and shows any inheritance relationships among the attributes. The **LDIF View** shows the equivalent schema definition in ASN.1 format, which includes the proper text spacing required for each schema element.

Default Directory Server Schema Files

The PingDirectory Server stores its schema as a set of LDIF files for a Directory Server instance in the `<server-root>/config/schema` directory. The Directory Server reads the schema files in alphanumeric order at startup, so that the 00-core.ldif file is read first, then 01-pwpolicy.ldif, and then the rest of the files. Custom schema files should be named so that they are loaded in last. For example, custom schema elements could be saved in a file labelled 99-user.ldif, which loads after the default schema files are read at startup.

The Directory Server then uses the schema definitions to determine any violations that may occur during add, modify, or import requests. Clients applications check the schema (i.e., matching rule definitions) to determine the assertion value algorithm used in comparison or search operations.

The default set of schema files are present at installation and should not be modified. Modifying the default schema files could result in an inoperable server.

The schema files have the following descriptions:

Table 33: Default Schema Files

Schema Files	Description
00-core.ldif	Governs the Directory Server's core functions.
01-pwpolicy.ldif	Governs password policies.
02-config.ldif	Governs the Directory Server's configuration.
03-changelog.ldif	Governs the Directory Server's change log.
03-rfc2713.ldif	Governs Java objects.
03-rfc2714.ldif	Governs Common Object Request Broker Architecture (CORBA) object references.
03-rfc2739.ldif	Governs calendar attributes for vCard.
03-rfc2926.ldif	Governs Server Location Protocol (SLP) mappings to and from LDAP schemas.
03-rfc2985.ldif	Governs PKCS #9 public-key cryptography.
03-rfc3112.ldif	Governs LDAP authentication passwords.
03-rfc3712.ldif	Governs printer services.
03-uddiv3.ldif	Governs web services registries of SOA components.
04-rfc2307bis.ldif	Governs mapping entities from TCP/IP and UNIX into X.500 entries.

Extending the Directory Server Schema

The PingDirectory Server stores its schema as LDIF files in the <server-root>/config/schema directory. At startup, the Directory Server reads the schema files once in alphanumeric order starting with 00-core.ldif and ending with any custom schema definition files, such as 99-user.ldif if present.

You can extend the schema to include additional customizations necessary for your Directory Server data using one of the following methods:

- **Using the Schema Editor**. This method is the easiest and quickest way to set up a schema definition and have it validated for the correct ASN.1 formatting. The Editor lets you define your schema properties, load your custom file, or perform a cut-and-paste operation on a new schema element. If any errors exist in the file, the Schema Editor generates an error message if the schema definitions do not pass compliance.
- **Using a Custom Schema File**. You can create a custom schema file with your new definitions using a text editor, save it as 99-user.ldif, and then import the file using the Schema Editor or the ldapmodify tool. You must name the custom LDIF file with a high two-digit number prefix, so that the Directory Server will read the file AFTER the core schema files are read at startup. For example, you can name the file, 99-myschema.ldif, etc. See the next section, General Tips on Extending the Schema to see the requirements for naming each file.
- **Using the Command Line**. If you have a small number of additions, you can extend the schema over LDAP and from the command line using the ldapmodify tool. The Directory Server writes the new schema changes to

a file `99-user.ldif` in the `<server-root>/config/schema` directory. However, this method can be cumbersome as schema definitions require strict adherence to text spacing and white space characters.

General Tips on Extending the Schema

You should consider the following points when extending the schema:

- Never modify the default schema files as doing so could damage the Directory Server's processing capabilities.
- Define all attributes first before they can be used in an object class. If you are using the Schema Editor to add new schema elements, then you can use the Quick Add Attributes option when defining new objectclasses.
- Define the parent object class first before creating object classes that inherit from the parent.
- The schema file naming syntax requires that custom schema files must begin with exactly two digits followed by a non-digit character, followed by a zero or more characters and ending with ".ldif". Note that the two digits do not need to be followed by a dash ("-"). Any files that do not meet this criteria will be ignored and either a NOTICE or SEVERE_WARNING message will be logged.

  Any file in the `<server-root>/config/schema` directory with a name that starts with "." or with a name that ends with a tilde ('~'), ".swp", or ".tmp" will generate a NOTICE message indicating that temporary files will be ignored. Any other file that does not meet the naming criteria will generate a SEVERE_WARNING message indicating that it will be ignored.
- Define custom attributes and object classes in one file. Typically, this file will be the `99-user.ldif`. You can specify a different file name to which the Directory Server writes using the X-SCHEMA-FILE element and the file name in the definition. For example:

```
add: attributeTypes attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.1
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example'
  X-SCHEMA-FILE '99-custom.ldif' )
```
- Pay special attention to the white space characters in the schema definitions, where WSP means zero or more space characters, and SP means one or more space characters. The LDIF specification states that LDIF parsers should ignore exactly one space at the beginning of each continuation line, since continuation lines must begin with a space character. Thus, if you define a new schema definition with each keyword on a separate continuation line, you should add two spaces before an element keyword to be safe. For example, the following attribute definition has two spaces before the keywords: NAME, SUP, and X-ORIGIN.

```
attributeTypes: ( 2.5.4.32 NAME 'owner' SUP distinguishedName X-ORIGIN 'RFC
 4519' )
```
- In a replicated topology, any new schema additions will be replicated to other replication servers to their respective Schema backend. The additions will be written to the file specified by the X-SCHEMA-FILE extension or written to `99-user.ldif` if no file is specified.

Managing Attribute Types

An attribute type determines the important properties related to an attribute, such as specifying the matching and syntax rules used in value comparisons. An attribute description consists of an attribute type and a set of zero or more options. Options are short, case-insensitive text strings that differentiate between attribute descriptions. For example, the LDAPv3 specification defines only one type of option, the tagging option, which can be used to tag language options, such as `cn;lang-de;lang-sp` or binary data, such as `userCertificate;binary`. You can also extend the schema by adding your own attribute definitions.

Attributes have the following properties:

- Attributes can be user attributes that hold information for client applications, or operational attributes that are used for administrative or server-related purposes. You can specify the purpose of the attribute by the `USAGE` element.
- Attributes are multi-valued by default. Multi-valued means that attributes can contain more than one value within an entry. Include the `SINGLE-VALUE` element if the attribute should contain at most one value within an entry.
- Attributes can inherit properties from a parent attribute as long as they both have the same USAGE, and the child attribute has the same `SYNTAX` or its `SYNTAX` allows values which are a subset of the values allowed by the `SYNTAX` of the parent attribute. For example, the surname (`sn`) attribute is a child of the name attribute.

Attribute Type Definitions

New attribute types do not require server code extensions if the provided matching rules and attribute syntaxes are used in the definitions. Administrators can create new attributes using the Schema Editor, which stores the definition in a file in the `<server-root>/config/schema` directory. See Extending the Directory Server Schema for more information.

The formal specification for attribute types is provided in RFC 4512, section 4.1.2 as follows:

```
AttributeTypeDescription = "(" wsp; Left parentheses followed by a white space
numericoid                        ; Required numeric object identifier
[ sp "NAME" sp qdescrs ]          ; Short name descriptor as alias for the OID
[ sp "DESC" sp qdstring ]         ; Optional descriptive string
[ sp "OBSOLETE" ]                 ; Determines if the element is active
[ sp "SUP" sp oid ]               ; Specifies the supertype
[ sp "EQUALITY" sp oid ]          ; Specifies the equality matching rule
[ sp "ORDERING" sp oid ]          ; Specifies ordering matching rule
[ sp "SUBSTR" sp oid ]            ; Specifies substrings matching rule
[ sp "SYNTAX" sp oidlen ]         ; Numeric attribute syntax with minimum
 upper bound
                                  ;  length expressed in {num}
[ sp "SINGLE-VALUE" ]             ; Specifies if the attribute is single
 valued in
                                  ;  the entry
[ sp "COLLECTIVE" ]               ; Specifies if it is a collective attribute
[ sp "NO-USER-MODIFICATION" ]     ; Not modifiable by external clients
[ sp "USAGE" sp usage ]           ; Application usage
extensions wsp ")"                ; Extensions followed by a white space and
 ")"

usage = "userApplications" /      ; Stores user data
  "directoryOperation" /          ; Stores internal server data
  "distributedOperation" /        ; Stores operational data that must be
 synchronized
                                  ; across servers
  "dSAOperation"                  ; Stores operational data specific to a
 server and
                                  ; should not be synchronized across servers
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /             ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /        ; Specifies which schema file contains the definition
"X-APPROX" /             ; Specifies the approximate matching rule
"X-ALLOWED-VALUE" /      ; Explicitly specifies the set of allowed values
"X-VALUE-REGEX" /        ; Specifies the set of regular expressions to compare
 against
                         ;  attribute values to determine acceptance
"X-MIN-VALUE-LENGTH" /   ; Specifies the minimum character length for
 attribute values
"X-MAX-VALUE-LENGTH" /   ; Specifies the maximum character length for
 attribute values
```

```
"X-MIN-INT-VALUE" /      ; Specifies the minimum integer value for the
 attribute
"X-MAX-INT-VALUE" /      ; Specifies the maximum integer value for the
 attribute
"X-MIN-VALUE-COUNT" /    ; Specifies the minimum number of allowable values
 for the
                         ;  attribute
"X-MAX-VALUE-COUNT" /    ; Specifies the maximum number of allowable values
 for the
                         ;  attribute
"X-READ-ONLY"            ; True or False. Specifies if the file that contains
 the
                         ;  schema element is marked as read-only in the
 server
                         ;  configuration.
```

Basic Properties of Attributes

The Basic Properties section displays the standard elements in schema definition.

Table 34: Basic Properties of Attributes

Attributes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the attribute and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Syntax	Specifies the attribute syntax used. For example, the `userPassword` attribute uses the User Password Syntax whereas the `authPassword` attribute uses the Authentication Password Syntax.
Parent	Specifies the schema definition’s parent or supertype if any. The analogous LDIF equivalent is "SUP".
Multivalued	Specifies if the attribute can appear more than once in its containing object class.
Required By Class	Specifies any object classes that require the attribute.
Allowed By Class	Specifies any object classes that can optionally use the attribute.
Value Restrictions	Specifies any restriction on the value of the attribute.

The Extra Properties section provides additional auxiliary information associated with the attribute.

Table 35: Basic Properties of Attributes

Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element. For example, NAME (’sn’ ’surname’).
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Stored in File	Specifies the schema file that stores the definition in the `<server-root>/config/schema` folder.

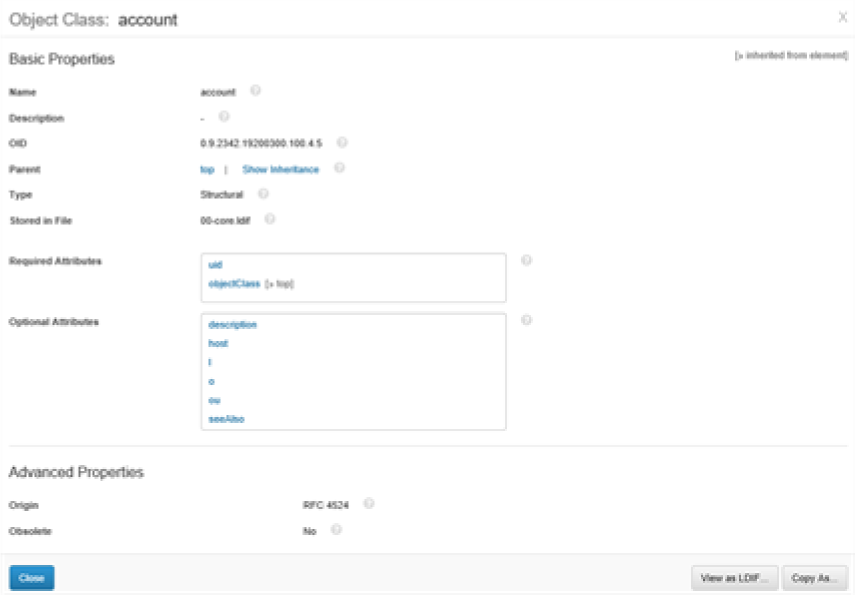
Attributes	Description
Usage	Specifies the intended use of the attribute. Choices are the following: userApplications directoryOperation distributedOperation dSAOperation
User-Modifiable	Specifies if the attribute can be modified by an authorized user.
Obsolete	Specifies if the schema definition is obsolete or not.
Matching Rules	Specifies the associated matching rules for the attribute.

Viewing Attributes

The Schema Editor displays all of the attribute types on your directory server instance. It shows the basic properties that are required elements plus the extra properties that are allowed within the attribute definition.

To View Attribute Types Using the Schema Editor

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is also running.
2. On the main menu, click **Schema**.
3. In the Administrative Console **Schema Editor**, click the **Attribute Types** tab.
4. Click a specific attribute to view its definition. In this example, click the `account` attribute. In the **Object Class** window, view the attribute properties.



Object Class: account

Basic Properties

[» inherited from element]

Name: account

Description: -

OID: 0.9.2342.19200300.100.4.5

Parent: top | Show Inheritance

Type: Structural

Stored in File: 00-core.ldif

Required Attributes: uid, objectClass [» top]

Optional Attributes: description, host, l, o, ou, seeAlso

Advanced Properties

Origin: RFC 4524

Obsolete: No

5. Click the **View as LDIF** button to see the equivalent attribute definition in ASN.1 format.

To View Attribute Types over LDAP

- Use `ldapsearch` to view a multi-valued operational attribute `attributeTypes`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" attributeTypes
```

To View a Specific Attribute Type over LDAP

- Use `ldapsearch` with the --dontWrap option and use the `grep` command to search for a specific attribute.

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base --dontWrap "(objectclass=*)" \
  attributeTypes | grep 'personalTitle'
```

Creating a New Attribute over LDAP

The following section shows how you can add the schema element from the previous section over LDAP. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting.

To Add an New Attribute to the Schema over LDAP

1. Create an LDIF file with the new attribute definition using a text editor. Save the file as `myschema.ldif`.

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ALLOWED-VALUES ’Y’ ’N’ ’y’ ’n’
  X-ORIGIN 'Directory Server Example' )
```

2. Use `ldapmodify` to add the attribute.

```
$ bin/ldapmodify --filename myschema.ldif
```

3. Verify the addition by displaying the attribute using `ldapsearch`.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  --dontwrap "(objectclass=*)" attributeTypes | grep 'contractorStatus'
```

4. You can view the custom schema file at `<server-root>/config/schema/99-user.ldif`. You should see the following:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( contractorStatus-OID
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Ping Identity Directory Server Example' )
```

To Add Constraints to Attribute Types

- The Directory Server provides attribute type extensions that constrain the values for the associated attribute using the `DirectoryString` attribute syntax. The following schema definition includes two `attributeType` definitions for `myAttr1` and `myAttr2`. The first definition constrains the values for the attribute `myAttr1` to ’foo’, ’bar’, ’baz’. The second definition constrains the minimum allowable length for `myAttr2` to 1 and the maximum allowable length to 5.

```
attributeTypes: (1.2.3.4
  NAME ’myAttr1’
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ALLOWED-VALUES ( ’foo’ ’bar’ ’baz’ ))
attributeTypes: ( 1.2.3.5
  NAME ’myAttr2’
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-MIN-VALUE-LENGTH ’1’
  X-MAX-VALUE-LENGTH ’5’ )
```

Managing Object Classes

Object classes are sets of related information objects that form entries in a Directory Information Tree (DIT). The Directory Server uses the schema to define these entries, to specify the position of the entries in a DIT, and to control the operation of the server. You can also extend the schema by adding your own schema definitions.

Object classes have the following general properties:

- Object classes must have a globally unique name or identifier.
- Object classes specify the required and allowed attributes in an entry.
- Object classes can inherit the properties and the set of allowed attributes from its parent object classes, which may also be part of a hierarchical chain derived from the top abstract object class.
- Object classes that are defined in the PingDirectory Server can be searched using the `objectClasses` operational attribute. The Directory Server also has a special entry called the subschema subentry, which provides information about the available schema elements on the server.

Object Classes Types

Based on RFC 4512, object classes can be a combination of three different types:

- **Abstract object classes** are used as the base object class, from which structural or auxiliary classes inherit its properties. This inheritance is a one-way relationship as abstract object classes cannot be derived from structural or auxiliary classes. The most common abstract object class is `top`, which defines the highest level object class in a hierarchical chain of object classes.
- **Structural object classes** define the basic attributes in an entry and define where an entry can be placed in a DIT. All entries in a DIT belong to one structural object class. Structural object classes can inherit properties from other structural object classes and from abstract object classes to form a chain of inherited classes. For example, the `inetOrgPerson` structural object class inherits properties from the `organizationalPerson` structural class, which inherits from another object class, person.
- **Auxiliary object classes** are used together with structural object classes to define additional sets of attributes required in an entry. The auxiliary object class cannot form an entry alone but must be present with a structural object class. Auxiliary object classes cannot derive from structural object classes or vice-versa. They can inherit properties from other auxiliary classes and from abstract classes.

Object Class Definition

New object classes can be specified with existing schema components and do not require additional server code extensions for their implementation. Administrators can create new object classes using the Schema Editor, which manages schema in the `<server-root>/config/schema` directory. See Extending the Directory Server Schema for more information.

The object class definition is defined in RFC 4512, section 4.1.1, as follows::

```
ObjectClassDescription = "(" wsp; Left parenthesis followed by a white space
numericoid                      ; Required numeric object identifier
[ sp "NAME" sp qdescrs ]        ; Short name descriptor as alias for the OID
[ sp "DESC" sp qdstring ]       ; Optional descriptive string
[ sp "OBSOLETE" ]               ; Determines if the element is inactive
[ sp "SUP" sp oid ]                ; Specifies the direct superior object class
[ sp kind ]                        ; abstract, structural (default), auxiliary
[ sp "MUST" sp oids ]              ; Required attribute types
[ sp "MAY" sp oids ]               ; Allowed attribute type
extensions wsp ")"                 ; Extensions followed by a white space and
 ")"

usage = "userApplications" /       ; Stores user data
  "directoryOperation" /           ; Stores internal server data
  "distributedOperation" /         ; Stores operational data that must be
 synchronized
                                   ;   across servers
  "dSAOperation"                   ; Stores operational data specific to a
 server and
                                   ;   should not be synchronized across
 servers
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /              ; Specifies where the object class is defined
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the definition
"X-READ-ONLY"             ; True or False. Specifies if the file that contains
                          ;   the schema element is marked as read-only
                          ;   in the server configuration.
```

Note: Although RFC 4512 allows multiple superior object classes, the PingDirectory Server allows at most one superior object class, which is defined by the SUP element in the definition.

Basic Object Class Properties

The Basic Properties section displays the standard elements in schema definition.

Table 36: Basic Properties of Attributes

Attributes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the object class and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Parent	Specifies the schema definition’s hierarchical parent or superior object class if any. An object class can have one parent. The analogous LDIF equivalent
Type	Specifies the type of schema definition: abstract, structural, or auxiliary. The analogous LDIF equivalent is "ABSTRACT", "STRUCTURAL", or "AUX
Required Attributes	Specifies any required attributes with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MUST".
Optional Attributes	Specifies any optional attributes that could be used with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MAY".

The Extra Properties section provides additional auxiliary information associated with the object class.

Table 37: Basic Properties of Attributes

Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element although most object classes do not have aliases.
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Obsolete	Specifies if the schema definition is obsolete or not.
Stored in File	Specifies the schema file that stores the definition in the `<server-root>/config/schema` folder.

Viewing Object Classes

You can view the object classes on your Directory Server by using the Administrative Console Schema Editor, over LDAP using the `ldapsearch` tool, or some third party tool. The Schema Editor displays all of the object classes on the directory server instance. It shows the basic properties that are required elements and the extra properties that are allowed within the object class.

To View Object Classes over LDAP

- Use `ldapsearch` tool to view a multi-valued operational attribute, `objectClasses`, which publishes the object class definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
--dontWrap "(objectclass=*)" objectClasses
```

```
dn: cn=schema
objectClasses: ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectClass X-ORIGIN 'RFC
 4512' )
objectClasses: ( 2.5.6.1 NAME 'alias' SUP top STRUCTURAL MUST
 aliasedObjectName
  X-ORIGIN 'RFC 4512' )
objectClasses: ( 2.5.6.2 NAME 'country' SUP top STRUCTURAL MUST c
  MAY ( searchGuide $ description ) X-ORIGIN 'RFC 4519' )
...(more output)...
```

Managing an Object Class over LDAP

The following section shows how you can manage an object class schema element over LDAP by adding a new attribute element to an existing object class. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting. The following example procedure adds an attribute, `contractorAddress`, to the custom schema file, then adds it to the `contractor` objectclass.

To Manage an Object Class over LDAP

1. Assume that you have defined the `contractorAddress` attribute, create an LDIF file called `contractorAddress-attr.ldif` with the following content:

```
dn: cn=schema
changetype: modify

add: attributeTypes attributeTypes: ( contractor-OID NAME
 'contractorAddress'
```

```
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'user defined'
  X-SCHEMA-FILE ’98-custom-schema.ldif’ )
  X-ORIGINS ’user defined’
  X-SCHEMA-FILE ’98-custom-schema.ldif’ )
```

2. Add the attribute using `ldapmodify`.

```
$ bin/ldapmodify --filename contractorAddress-attr.ldif
```

3. Next, create an LDIF file to modify the contractor objectclass to allow this attribute. When doing this, you are just re-adding the updated `objectClass` and the Directory Server will handle the proper replacement of the existing object class with the new one. Create a file called `contractor-oc.ldif`. Make sure that the lines are not wrapped, the `objectClasses` line should be one continuous line.

```
dn:cn=schema
changetype: modify
add: objectClasses
objectClasses: ( contractor-OID NAME ’contractor’
  DESC ’Contractor status information
  SUP top
  AUXILIARY MAY ( contractorStatus $ contractorAgency $ contractorAddress )
  X-ORIGIN ’Directory Server Example’
  X-SCHEMA-FILE ’98-custom-schema.ldif’ )
```

4. Update the `objectClass` using `ldapmodify` as follows:

```
$ bin/ldapmodify --filename contractor-oc.ldif
```

5. These schema changes will be replicated to all servers in the replication topology. Verify the change by looking at the `config/schema/98-custom-schema.ldif` file on the other servers in the replication topology to ensure that the changes are present.
6. If you need to add an index for this attribute, you can do so by using the `dsconfig` command-line utility. You will need to do this on each server in your topology unless you have server configuration groups set up. See Configuring Server Groups for more information.

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name contractorAddress --set index-type:equality
```

7. Rebuild the index online. This will not affect other indexes or entries since there is no currently existing data for this attribute on any entry.

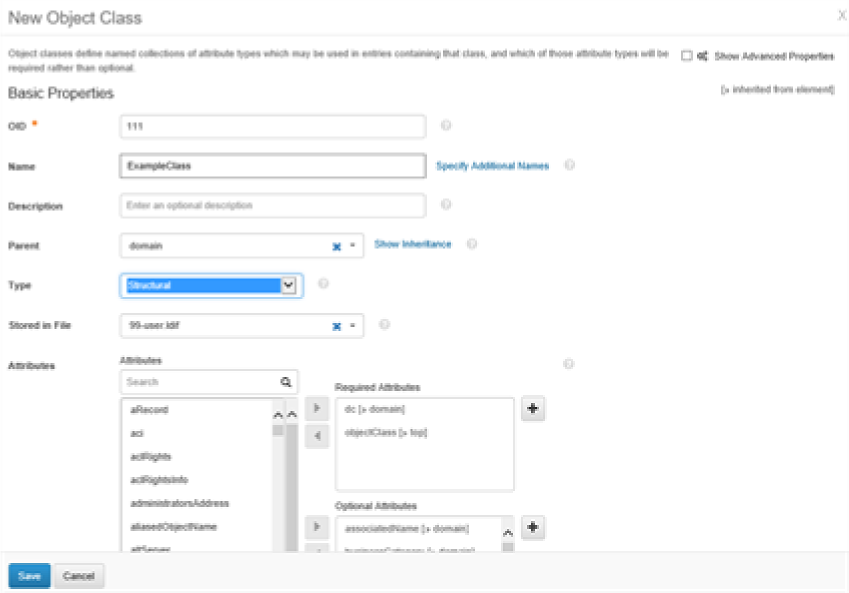
```
$ bin/rebuild-index --baseDN dc=example,dc=com --index contractorAddress
```

Creating a New Object Class Using the Schema Editor

The procedures to create a new object class are similar to that of creating a new attribute. Make sure that any attributes that are part of the new object class are defined prior to defining the object class.

To Create a New Object Class Using the Schema Editor

1. Start the Administrative Console.
2. On the main menu, click **LDAP Schema**.
3. On the **Schema Editor**, click the **Object Classes** tab, and then click **New** located in the bottom left of the window.
4. Enter the properties for the new objectclass. In the **Attributes** box, filter the types of attributes required for the new object class. Click the right arrow to move it into the **Required** or the **Optional** box. All custom attributes appear at the bottom of the list in the **Attributes** box.



New Object Class

Object classes define named collections of attribute types which may be used in entries containing that class, and which of those attribute types will be required rather than optional.

□ Show Advanced Properties

Basic Properties

[› inherited from element]

OID *: 111

Name: ExampleClass

Description: Enter an optional description

Parent: domain

Type: Structural

Stored in File: 99-user.ldif

Attributes

Required Attributes: dc [› domain], objectClass [› top]

Optional Attributes: associatedName [› domain], businessCategory [› domain]

Extending the Schema Using a Custom Schema File

You can add new attributes and object classes to your Directory Server schema by creating a custom schema file. You can import the file using the Schema Editor, over LDAP using the `ldapmodify` tool, or from the command line. Make sure to define the attributes first, then define the object classes.

To Extend the Schema Using a Custom Schema File

1. Create an LDIF file with the new attribute extensions using a text editor.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example' )
attributeTypes: ( contractorAgency-OID NAME 'contractorAgency'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{256}
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'PingDirectory Server Example' )
```

2. In the same LDIF file, add a new object class definition after the attribute types. In this example, create an auxiliary object class, `contractor`, that alone cannot be used as an entry. The object class will be used to add supplemental information to the `inetOrgPerson` structural object class. The attributes are all optional for the new object class.

```
objectClasses: ( contractor-OID
  NAME 'contractor'
  DESC 'Contractor status information'
  SUP top
  AUXILIARY
  MAY ( contractorStatus $ contractorAgency )
  X-ORIGIN 'PingDirectory Server Example' )
```

3. Save the file as `99-auxobjclass.ldif`. and place it in the `<server-root>/config/schema` directory.
4. At this stage, the schema extensions are not loaded into the Directory Server yet. You have four options to load them:

   - Create a task that loads the new extensions into the schema. We create a task labelled with the ID "add-schema-99-auxobjclass" and add it using `ldapmodify`. The server does not need to be restarted using this method.

     ```
     dn: ds-task-id=add-schema-99-auxobjclass,cn=Scheduled Tasks,cn=tasks
     objectClass: top
     objectClass: ds-task
     objectClass: ds-task-add-schema-file
     ds-task-id: add-schema-99-auxobjclass
     ds-task-class-name:
      com.unboundid.directory.server.tasks.AddSchemaFileTask
     ds-task-schema-file-name: 99-auxobjclass.ldif
     ```
   - Import the schema file using the Administrative Console Schema Editor. You do not need to restart the server when using this method.
   - Place the `99-auxobjclass.ldif` file in the `<server-root>/config/schema` directory and restart the Directory Server. The schema file is read at startup.
   - Add the schema file using `load-ldap-schema-file`. You do not need to restart the server when using this method.

     ```
     $ bin/load-ldap-schema-file --schemaFile config/schema 99-auxobjclass.ldif
     ```
5. Verify the addition by displaying the attribute using `ldapsearch`.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.9)" contractorStatus
   ```

   ```
   dn: uid=user.9,ou=People,dc=example,dc=com
   contractorStatus: TRUE
   ```

Managing Matching Rules

Matching rules determine how clients and servers compare attribute values during LDAP requests or operations. They are also used in evaluating search filter elements including distinguished names and attributes. Matching rules are defined for each attribute based on EQUALITY (e.g., two attributes are equal based on case, exact match, etc.), SUBSTR (e.g., assertion value is a substring of an attribute), and ORDERING (e.g., greater than or equal, less than or equal, etc.) properties.

Note: The PingDirectory Server supports an `APPROXIMATE` matching rule that compares similar attributes based on fuzzy logic. Thus, attributes that are similar or "sound-like" each other are matched. For example, "petersen" would match "peterson".

Matching Rule Definition

New matching rules require additional server code extensions to be implemented on the PingDirectory Server. If you need new matching rules, contact your authorized support provider for assistance.

The formal specification for attribute types is provided in RFC 4512, section 4.1.3 as follows:

```
MatchingRuleDescription = "(" wsp    ; Left parentheses followed by a white
 space
numericoid                           ; Required numeric object identifier
 identifying
                                     ;   this matching rule
[ sp "NAME" sp qdescrs               ; Short name descriptor
[ sp "DESC" sp qdstring              ; Description
```

```
[ sp "OBSOLETE" ]                  ; Specifies if the rule is inactive
sp "SYNTAX" sp numericoid          ; Assertion syntax
extensions wsp ")"                 ; Extensions followed by a white space
 and ")"
```

Default Matching Rules

The PingDirectory Server provides a large set of matching rules, which support a variety of applications. The default matching rules available for the Directory Server are listed in the table below for each matching rule type: Equality, Substring, Ordering, and Approximate matches.

Table 38: Default Matching Rules

Matching Rule/OID	Attribute Syntax/OID	Description
uuidMatch/ 1.3.6.1.1.16.2	UUID/ 1.3.6.1.1.16.1	Compares an asserted UUID with a stored UUID attribute value for equality. RFC 4530.
uuidOrderingMatch/ 1.3.6.1.1.16.3	UUID/ 1.3.6.1.1.16.1	Compares the collation order of an asserted UUID with a stored UUID attribute value for ordering RFC 4530.
caseExactIA5Match/ 1.3.6.1.4.1.1466.109.114.1	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. RFC 4517.
caseIgnoreIA5Match/ 1.3.6.1.4.1.1466.109.114.2	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreIA5SubstringsMatch/ 1.3.6.1.4.1.1466.109.114.3	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
authPasswordExactMatch/ 1.3.6.1.4.1.4203.1.2.2	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password exact matching rule.
authPasswordMatch/ 1.3.6.1.4.1.4203.1.2.3	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password matching rule.
ds-mr-double-metaphone-approx/ 1.3.6.1.4.1.30221.1.4.1	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Syntax based on the phonetic Double Metaphone algorithm for approximate matching.
ds-mr-user-password-exact/ 1.3.6.1.4.1.30221.1.4.2	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password exact matching rule.
ds-mr-user-password-equality/ 1.3.6.1.4.1.30221.1.4.3	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password equality matching rule.
historicalCsnOrderingMatch/ 1.3.6.1.4.1.30221.1.4.4	1.3.6.1.4.1.30221.1.3.5	Compares the collation order of a historical change sequence number with a historical CSN attribute value.

Matching Rule/OID	Attribute Syntax/OID	Description
caseExactIA5SubstringsMatch/ 1.3.6.1.4.1.30221.1.4.902	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. RFC 4517.
compactTimestampMatch/ 1.3.6.1.4.1.30221.2.4.1	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compact Timestamp matching rule.
compactTimestampOrderingMatch/ 1.3.6.1.4.1.30221.2.4.2	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compares the collation order of a compact timestamp number with an attribute value of Compact Timestamp syntax.
objectIdentifierMatch/ 2.5.13.0	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Compares an asserted value with an attribute value of OID syntax. RFC 4517.
distinguishedNameMatch/ 2.5.13.1	DN/ 1.3.6.1.4.1.1466.115.121.1.12	Compares an asserted value with an attribute value of DN syntax. Spaces around commas and semicolons are ignored. Spaces around plus and equal signs around RDN components are ignored. RFC 4517.
caseIgnoreMatch/ 2.5.13.2	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreOrderingMatch/ 2.5.13.3	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
caseIgnoreSubstringsMatch/ 2.5.13.4	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring value with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
caseExactMatch/ 2.5.13.5	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value of Directory String syntax. RFC 4517.
caseExactOrderingMatch/ 2.5.13.6	Directory String 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. RFC 3698
caseExactSubstringsMatch/ 2.5.13.7	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Directory String syntax. RFC 3698

Matching Rule/OID	Attribute Syntax/OID	Description
numericStringMatch/ 2.5.13.8	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares an asserted value with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringOrderingMatch/ 2.5.13.9	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares the collation order of the asserted string with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringSubstringsMatch/ 2.5.13.10	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
caseIgnoreListMatch/ 2.5.13.11	Postal Address/ 1.3.6.1.4.1.1466.115.121.1.41	Compares an asserted value with an attribute value which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreListSubstringsMatch/ 2.5.13.12	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares the asserted substring with an attribute value, which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 3698.
booleanMatch/ 2.5.13.13	Boolean/ 1.3.6.1.4.1.1466.115.121.1.7	Compares an asserted boolean value with an attribute value of BOOLEAN syntax. Returns true if the values are both TRUE or both FALSE. RFC 3698.
integerMatch/ 2.5.13.14	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares an asserted value with an attribute value of INTEGER syntax. RFC 4517.
integerOrderingMatch/ 2.5.13.15	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares the collation order of the asserted integer with an attribute value of Integer syntax. Returns true if the attribute value is less than the asserted value. RFC 3698.
bitStringMatch/ 2.5.13.16	Bit String/ 1.3.6.1.4.1.1466.115.121.1.6	Compares an asserted Bit String value with an attribute value of Bit String syntax. RFC 4517.
octetStringMatch/ 2.5.13.17	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares an asserted value with an attribute value of octet string

Matching Rule/OID	Attribute Syntax/OID	Description
		syntax using a byte-for-byte comparison. RFC 4517.
octetStringOrderingMatch/ 2.5.13.18	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares the collation order of the asserted octet string with an attribute value of Octet String syntax. Zero precedes a one bit. Shorter strings precede longer strings. RFC 3698.
octetStringSubstringsMatch/ 2.5.13.19	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
telephoneNumberMatch/ 2.5.13.20	Telephone Number/ 1.3.6.1.4.1.1466.115.121.1.50	Compares an asserted value with an attribute value of Telephone Number syntax. RFC 4517.
telephoneNumberSubstringsMatch/ 2.5.13.21	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted value with the substrings of an attribute value of Telephone Number String syntax. RFC 4517.
presentationAddressMatch/ 2.5.13.22	Presentation Address/ 1.3.6.1.4.1.1466.115.121.1.43	Compares an asserted value with an attribute value of Presentation Address syntax. RFC 4517.
uniqueMemberMatch/ 2.5.13.23	Name and Optional UID/ 1.3.6.1.4.1.1466.115.121.1.34	Compares an asserted value with an attribute value of Unique Member syntax. RFC 4517.
protocolInformationMatch/ 2.5.13.24	Protocol Information/ 1.3.6.1.4.1.1466.115.121.1.42	Compares an asserted value with an attribute value of Protocol Information syntax. RFC 4517.
generalizedTimeMatch/ 2.5.13.27	Generalized Time/ 1.3.6.1.4.1.1466.115.121.1.24	Compares an asserted value with an attribute value of Generalized Time syntax. RFC 4517.
generalizedTimeOrderingMatch/ 2.5.13.28	Generalized Time 1.3.6.1.4.1.1466.115.121.1.24	Compares the collation order of the asserted string with an attribute value of Generalized Time String syntax and case is ignored. RFC 4517.
integerFirstComponentMatch/ 2.5.13.29	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Equality matching rules for subschema attributes between an Integer syntax and the value syntax. RFC 4517.
objectIdentifierFirstComponentMatch/ 2.5.13.30	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Equality matching rules for subschema attributes between an OID syntax and the value syntax. RFC 4517.
directoryStringFirstComponentMatch/ 2.5.13.31	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted Directory String value with an attribute value of type SEQUENCE whose first component is mandatory

Matching Rule/OID	Attribute Syntax/OID	Description
		and of type Directory String. Returns true if the attribute value has a first component whose value matches the asserted Directory String using the rules of caseIgnoreMatch. RFC 3698.
wordMatch/ 2.5.13.32	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted word with any word in the attribute value for equality RFC 3698.
keywordMatch/ 2.5.13.33	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with any keyword in the attribute value for equality. RFC 3698.

Basic Matching Rule Properties

The Properties section displays the standard elements in a matching rule schema definition.

Table 39: Basic Properties of Matching Rules

Attributes	Description
Name	Specifies the descriptive and unique name of the element.
Description	Specifies an optional definition that describes the matching rule. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Type	Specifies the type of type of matching rule: Equality, Ordering, Substring, or Approximate.
Syntax	Specifies the matching rule syntax.
Used by Attributes	Specifies any attributes that use the corresponding matching rule.

Viewing Matching Rules

You can view the matching rules on your Directory Server by using the Schema Editor on the Administrative Console, over LDAP using the `ldapsearch` tool, or some third party tool.

The Schema Editor displays all of the matching rules on your Directory Server instance. It shows the basic properties that are allowed within the matching rule.

To View Matching Rules Over LDAP

- Use `ldapsearch` to view a multi-valued operational attribute, `matchingRules`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" matchingRules
```

Managing Attribute Syntaxes

The attribute type definition has a SYNTAX element, or attribute syntax, that specifies how the data values for the attribute are represented. The syntax can be used to define a large range of data types necessary for client applications. An attribute syntax uses the Abstract Syntax Notation One (ASN.1) format for its definitions.

Attribute Syntax Definition

New attribute syntaxes require additional code to be implemented on the PingDirectory Server. If you need new syntax definitions, contact your authorized support provider for assistance.

The formal specification for attribute types is provided in RFC 4512, section 4.1.5 as follows:

```
SyntaxDescription = "(" wsp
numericoid              ; Object identifier
[ sp "DESC" sp qdstring ] ; Description
extensions wsp ")"       ; Extensions followed by a white space and ")"
```

Default Attribute Syntaxes

The PingDirectory Server supports a large set of Attribute Syntax rules for applications. The default Attribute Syntax rules available for the directory server are listed in the table below.

Table 40: Default Attribute Syntaxes

LDAP Syntax	OID	Description
UUID	1.3.6.1.1.16.1	128-bit (16 octets) Universally Unique Identifier (UUID) used for Uniform Resource Names as defined in RFC 4122. For example, a4028c1a-f36e-11da-ba1a-04112154bd1e.
Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3	Syntax for the AttributeTypeDescription rule based on RFC 4517.
Binary	1.3.6.1.4.1.1466.115.121.1.5	Strings based on Basic Encoding Rules (BER) or Distinguished Encoding rules (DER). For example, an X.509 digital certificate or LDAP messages are BER encoded.
Bit String	1.3.6.1.4.1.1466.115.121.1.6	Sequence of binary digits based on RFC 4517. For example, '0010111'B.
Boolean	1.3.6.1.4.1.1466.115.121.1.7	TRUE or FALSE.
Certificate	1.3.6.1.4.1.1466.115.121.1.8	BER/DER-encoded octet strings based on an X.509 public key certificate as defined in RFC 4523.
Certificate List	1.3.6.1.4.1.1466.115.121.1.9	BER/DER-encoded octet string based on an X.509 certificate revocation list as defined in RFC 4523.
Certificate Pair	1.3.6.1.4.1.1466.115.121.1.10	BER/DER-encoded octet string based on an X.509 public key certificate pair as defined in RFC 4523.
Country String	1.3.6.1.4.1.1466.115.121.1.11	Two character country code specified in ISO 3166. For example, US, CA, etc.

LDAP Syntax	OID	Description
DN	1.3.6.1.4.1.1466.115.121.1.12	Distinguished name of an entry as defined in RFC4514.
Delivery Method	1.3.6.1.4.1.1466.115.121.1.14	Sequence of services in preference order by which an entity receives messages as defined in RFC4517. For example, videotext $ telephone.
Directory String	1.3.6.1.4.1.1466.115.121.1.15	String of one or more characters from the Universal Character Set (UCS) using UCS Transformation Format 8 (UTF-8) encoding of the string.
DIT Content Rule Description	1.3.6.1.4.1.1466.115.121.1.16	DITContentRuleDescription as defined in RFC4517.
DIT Structure Rule Description	1.3.6.1.4.1.1466.115.121.1.17	DITStructureRuleDesciption as defined in RFC4517.
Enhanced Guide	1.3.6.1.4.1.1466.115.121.1.21	Combination of attribute types and filter operators to be used to construct search filters as defined in RFC4517. For example, person#(sn $EQ)#oneLevel.
Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22	Fax telephone number on the public switched telephone network as defined in RFC4517.
Fax	1.3.6.1.4.1.1466.115.121.1.23	Image generated using Group 3 fax process as defined in RFC4517.
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	String representing data and time as defined in RFC4517. YYYYMMDDHHMMSS[.|,fraction] [(+|-HHMM)|Z] For example, 201103061032, 201103061032-0500, or 201103061032Z ("Z" indicates Coordinated Universal Time).
Guide	1.3.6.1.4.1.1466.115.121.1.25	Attribute types and filter operators as defined in RFC4517.
IA5 String	1.3.6.1.4.1.1466.115.121.1.26	String of zero or more characters from the International Alphabet 5 (IA5) character set as defined in RFC4517.
Integer	1.3.6.1.4.1.1466.115.121.1.27	String representations of integer values. For example, the character string "1234" represents the number 1234 as defined in RFC4517.
JPEG	1.3.6.1.4.1.1466.115.121.1.28	Image in JPEG File Interchange Format (JFIF) as defined in RFC4517.
Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30	MatchingRuleDescription as defined in RFC4512.
Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31	Attribute types to which a matching rule is applied in an extensibleMatch search filter (RFC4511).
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	Distinguished name and an optional unique identifier that differentiates identical DNs as defined in RFC4517. For example, uid=jsmith,ou=People,dc=example,dc=com#'0111'B

LDAP Syntax	OID	Description
Name Form Description	1.3.6.1.4.1.1466.115.121.1.35	NameFormDescription as defined in RFC4512.
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	Sequence of one or more numerals and spaces as defined in RFC4517. For example, 14 848 929 102.
Object Class Description	1.3.6.1.4.1.1466.115.121.1.37	ObjectClassDescription as defined in RFC 4512.
OID	1.3.6.1.4.1.1466.115.121.1.38	Object identifier as defined in RFC 4512.
Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39	Specifies an electronic mailbox as defined in RFC 4517. For example, otherMailbox = google $ user@gmail.com
Octet String	1.3.6.1.4.1.1466.115.121.1.40	Sequence of zero or more octets (8-bit bytes) as defined in RFC4517.
Postal Address	1.3.6.1.4.1.1466.115.121.1.41	Strings of characters that form a multi-line address in a physical mail system. Each component is separated by a "$". For example, 1234 Main St.$Austin, TX 78744$USA.
Protocol Information	1.3.6.1.4.1.1466.115.121.1.42	Undefined.
Presentation Address	1.3.6.1.4.1.1466.115.121.1.43	String encoded OSI presentation address as defined in RFC 1278. For example, TELEX +00728722+RFC-1006+03+10.0.0.6
Printable String	1.3.6.1.4.1.1466.115.121.1.44	String of one or more printable ASCII alphabetic, numeric, and punctuation characters as defined in RFC 4517.
RFC3672 Subtree Specification	1.3.6.1.4.1.1466.115.121.1.45	Syntax based on subtree specification as defined as RFC 3672.
Supported Algorithm	1.3.6.1.4.1.1466.115.121.1.49	Octet string based on the LDAP-encoding for a supported algorithm value that results from the BER encoding of a SupportedAlgorithm ASN.1 value.
Telephone Number	1.3.6.1.4.1.1466.115.121.1.50	String of printable international telephone number representations in E.123 format as defined in RFC 4517. For example, +1 512 904 5525.
Teletex Terminal Identifier	1.3.6.1.4.1.1466.115.121.1.51	Identifier and telex terminal as defined in RFC 4517.
Telex Number	1.3.6.1.4.1.1466.115.121.1.52	String representing the telex number, country code, and answerback code as defined in RFC 4517. For example, 812374, ch, ehhg.
UTC Time	1.3.6.1.4.1.1466.115.121.1.53	Character string representing the data and time in UTC Time format as defined as RFC 4517: YYMMDDHHMM[SS][(+|-HHMM)|Z], where Z is the coordinated universal time. For example, 0903051035Z, 0903051035-0500.
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54	SyntaxDescription as defined in RFC4512.
Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58	Syntax for assertion values in an extensible match as defined in RFC 4517.

LDAP Syntax	OID	Description
Authentication Password Syntax	1.3.6.1.4.1.4203.1.1.2	Encoded password storage syntax as defined in RFC 3112. For example, the syntax specifies the storage scheme in brackets as follows: <storage-scheme>$<auth component>$<auth value> For example: SSHA$xdEZWRqgyJk=$egDEFDXvdeeEnXUEIDPnd39dkpe=
User Password Syntax	1.3.6.1.4.1.30221.1.3.1	Encoded password storage syntax as defined in RFC 2307. For example, the syntax specifies the storage scheme in brackets as follows: {SSHA}XaljOF0ii3fOwCrU1klgBpWFayqSYs+5W1pMnw==
Relative Subtree Specification	1.3.6.1.4.1.30221.1.3.2	Similar to the RFC 3672 subtree specification except it uses an LDAP search filter as the specification filter.
Absolute Subtree Specification	1.3.6.1.4.1.30221.1.3.3	Syntax for a subset of entries in a subtree based on RFC 3672.
Sun-defined Access Control Information	1.3.6.1.4.1.30221.1.3.4	Syntax for access control instructions used in Sun Directory Servers.
Compact Timestamp	1.3.6.1.4.1.30221.2.3.1	Syntax based on compact timestamp ISO 8601 format. For example, 20110306T102532.

Basic Attribute Syntax Properties

The Properties section displays the standard elements in an attribute syntax.

Table 41: Basic Properties of Attribute Syntaxes

Attributes	Description
Name	Specifies the descriptive and unique name of the element.
Description	Indicates an optional definition that describes the attribute syntax. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition.
Used by Attributes	Indicates any attributes that use the corresponding attribute syntax.

Viewing Attribute Syntaxes

You can view the attribute syntaxes on your Directory Server by using the Schema Editor on the Administrative Console, over LDAP using the `ldapsearch` tool, or some third party tool.

The Schema Editor displays all of the attribute syntaxes on your Directory Server instance. It shows the properties that are allowed within the attribute syntax.

To View Attribute Syntaxes Over LDAP

- Use `ldapsearch` to view the Directory Server's published list of attribute syntaxes using the multi-valued operational attribute, `ldapSyntaxes`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base "(objectclass=*)" ldapSyntaxes
```

Using the Schema Editor Utilities

The Schema Editor provides a **Schema Utilities** tab that enables importing new schema elements from a file and to checking schema compliance. If importing a schema file, the system automatically checks for compliance prior to the import. If the definition does not meet schema compliance, the system will display an error message. However, it is good practice to first check if your file is compliant with your schema prior to importing it.

To Check Schema Compliance Using the Schema Editor

1. Start the Administrative Console.
2. On the main menu, click **LDAP Schema**.
3. On the Schema Editor, click the **Schema Utilities** tab.
4. Click **Import Schema Elements** to read in an LDIF file, or copy-and-paste a new schema definition, and then click **Validate Entries**. If there is a problem, an error will be generated.

Modifying a Schema Definition

The Directory Server only allows schema definitions that are read-write to be edited. Schema elements indicated by the **Modifiable** column in the Schema Editor's tables can be modified.

To Modify a Schema Definition

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is running.
2. On the main menu, click **Schema**.
3. On the Schema Editor, click the **Object Classes** tab.
4. Select the object class that you want to modify, and then click **Actions -> Edit**. The Edit dialog box appears.
5. Make your changes, and then click **OK**.

Deleting a Schema Definition

The Directory Server only allows schema definitions that are read-write to be deleted. In general, those schema definitions in the **Custom** folder of the Schema Editor can be removed from the system. You should make sure that the schema element is not currently in use.

To Delete a Schema Definition

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is also running.
2. On the main menu, click **Schema**.
3. On the **Schema Editor**, click the **Object Classes** tab.
4. Select the object class that you want to remove, and then click **Actions -> Delete**.
5. On the **Confirmation** dialog, click **Yes** if you are sure that you want to delete the schema element.

Schema Checking

The PingDirectory Server provides full support for parsing all schema elements and provides access to all of its components. By default, the Directory Server enables schema checking for all operations, especially when importing data to the server or when modifying entries using the `ldapmodify` tool. Any schema violations will generate an error message to standard output.

To View the Schema Checking Properties

- Use `dsconfig` to view the schema checking property.

```
$ bin/dsconfig get-global-configuration-prop \
  --property check-schema
```

To Disable Schema Checking

Although not recommended, you can use the `dsconfig` tool to disable the schema checking. This feature only applies to public backends. Schema checking is enforced on private backends, such as changes to the Configuration, Schema, Task, and others. An admin action alert will be generated when attempting to disable schema checking using `dsconfig` interactive or non-interactive mode. The alert provides alternatives to disabling schema checking.

Run the `dsconfig` command and specify the `set-global-configuration-prop` subcommand to disable the `check-schema` property.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set check-schema:false
```

The system generates an admin action alert providing alternate options to disabling schema checking. Press **Enter** to continue the process or following one of the suggested tasks:

```
One or more configuration property changes require administrative action or
confirmation/notification.

Those properties include:

  *   check-schema: Schema checking should only be disabled as a last
 resort
      since disabling schema checking harms performance and can lead to
      unexpected behavior in the server as well as the applications that
      access it. There are less severe options for addressing schema issues:


  1.  Update the data to conform to the server schema.

  2.  Modify the server schema to conform to the data. Contact support
 before
      modifying the server's default schema.

  3.  Change the single-structural-objectclass-behavior property to allow
```

```
     entries to have no structural object class or multiple structural
object
     classes.

 4.  Change the invalid-attribute-syntax-behavior property to allow
attribute
     values to violate their attribute syntax.

 5.  Change the allow-zero-length-values property of the Directory String
     Attribute Syntax configuration to allow attributes with this syntax
to
     have a zero length value.

Continue?  Choose 'no' to return to the previous step (yes / no) [yes]:
```

Managing Matching Rule Uses

Matching Rule Use definitions map certain attribute types with a matching rule definition for extensible match filters. Extensible match filters allows clients to search using DN components, for example, (`ou:dn:=engineering`) or using an OID number, for example, (`cn:1.2.3.4:=Sam Carter`). The matching rule use attribute publishes those attribute types and matching rule combinations, which can be used in extensible match assertions.

Typically, you define a matching rule use that is not normally specified in the attribute type definition. You can create new matching rule uses from the existing schema definitions by adding a custom schema file in the `<server-root>/config/schema` directory.

Matching Rule Use Definitions

Matching Rule Use can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.4 as follows:

```
MatchingRuleUseDescription = "(" wsp
numericoid                   ; Object identifier
[ sp "NAME" sp qdescrs ]     ; Short name descriptor
[ sp "DESC" sp qdstring ]    ; Description
[ sp "OBSOLETE" ]            ; Specifies if the rule use is inactive
sp "APPLIES" sp oid          ; Attribute types
extensions wsp ")"           ; Extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the definition
"X-READ-ONLY"             ; True or False. Specifies if the file that contains
                          ;   the schema element is marked as read-only in the
                          ;   server configuration.
```

To View Matching Rule Uses

A matching rule use lists the attribute types that are suitable for use with an `extensibleMatch` search filter.

- Use `ldapsearch` to view the Directory Server's published list of matching rule uses using the multi-valued operational attribute, `matchingRuleUse`, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" matchingRuleUse
```

Managing DIT Content Rules

DIT Content Rules provide a way to precisely define what attributes may be present in an entry, based on its structural object class, without specifically creating a new object class definition. The DIT content rules can define the mandatory and optional attributes that entries contain, the set of auxiliary object classes that entries may be part of, and any optional attributes from the structural and auxiliary object classes that are prohibited from being present in the entries.

DIT Content Rule Definitions

DIT Content Rules can be specified with existing schema components and do not require additional code for its implementation. On the PingDirectory Server, only one DIT Content Rule may be defined for an entry in the structural object class.

The formal specification for attribute types is provided in RFC 4512, section 4.1.6 as follows:

```
DITContentRuleDescription = "(" wsp
numericoid                  ; Object identifier of the structural object class
 the rule applies to
[ sp "NAME" sp qdescrs ]    ; Short name descriptor
[ sp "DESC" sp qdstring ]   ; Description
[ sp "OBSOLETE" ]           ; Specifies if the rule is inactive
[ sp "AUX" sp oids ]        ; List of allowed auxiliary object classes
[ sp "MUST" sp oids ]       ; List of required attributes
[ sp "MAY" sp oids ]        ; List of allowed attributes in the entry
[ sp "NOT" sp oids ]        ; List of prohibited attributes in the entry
extensions wsp ")"          ; Extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /             ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /        ; Specifies which schema file contains the definition
"X-READ-ONLY"            ; True or False. Specifies if the file that contains
                         ;   the schema element is marked as read-only in
                         ;   the server configuration.
```

To View DIT Content Rules

- Use `ldapsearch` to view a multi-valued operational attribute `dITContentRules`, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

  ```
  $ bin/ldapsearch --baseDN cn=schema --searchScope base \
    "(objectclass=*)" dITContentRules
  ```

Managing Name Forms

Name Forms define how entries can be named based on their structural object class. Specifically, name forms specify the structural object class to be named, as well as the mutually-exclusive set of required and allowed attributes to form the Relative Distinguished Names (RDNs) of the entries. Each structural object class may be associated with at most one name form definition.

Name Form Definitions

Name Forms can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.7.2 as follows:

```
NameFormDescription = "(" wsp
numericoid                    ; object identifier
[ sp "NAME" sp qdescrs ]      ; short name descriptor
[ sp "DESC" sp qdstring ]     ; description
[ sp "OBSOLETE" ]             ; not active
sp "OC" sp oid                ; structural object class
sp "MUST" SP oids             ; attribute types
[ sp "MAY" sp oids ]          ; attribute types
extensions wsp ")"            ; extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /                ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /           ; Specifies which schema file contains the definition
"X-READ-ONLY"               ; True or False. Specifies if the file that contains
                            ;   the schema element is marked as read-only in
                            ;   the server configuration.
```

To View Name Forms

- Use `ldapsearch` to view a multi-valued operational attribute `nameForms`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base "(objectclass=*)"
 nameForms
```

Managing DIT Structure Rules

DIT Structure Rules define which entries may be superior or subordinate to other entries in the DIT. Together with name forms, DIT Structure Rules determine how RDNs are added together to make up distinguished names (DNs). Because DITs do not have a global standard and are specific to a company's implementation, each DIT structure rule associates a name form with an object class and specifies each structure rule with an integer rule identifier, instead of an OID number. The identifier defines its relationship, either superior or subordinate, to another object class. If no superior rules are specified, then the DIT structure rule applies to the root of the subtree.

DIT Structure Rule Definition

DIT Structure Rules can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.7.1 as follows:

```
DITStructureRuleDescription = "(" wsp
ruleid                        ; object identifier
[ sp "NAME" sp qdescrs ]      ; short name descriptor
[ sp "DESC" sp qdstring ]     ; description
[ sp "OBSOLETE" ]             ; specifies if the rule is inactive
sp "FORM" sp oid              ; OID or name form with which the rule is
 associated
[ sp "SUP" ruleids ]          ; Superior rule IDs
extensions wsp ")"            ; extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /                ; Specifies where the rule is defined
"X-SCHEMA-FILE" /           ; Specifies which schema file contains the definition
"X-READ-ONLY"               ; True or False. Specifies if the file that contains
                            ;   the schema element is marked as read-only in
                            ;   the server configuration.
```

To View DIT Structure Rules

- Use `ldapsearch` to view a multi-valued operational attribute `dITStructureRules`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" dITStructureRules
```

Managing JSON Attribute Values

The PingDirectory Server supports a JSON object attribute syntax, which can be used for attribute types whose values are JSON objects. The syntax requires that each value of this type is a valid JSON object. The following is an example schema definition:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( jsonAttr1-OID NAME 'jsonAttr1' DESC 'test json attribute
 support' EQUALITY jsonObjectExactMatch SYNTAX 1.3.6.1.4.1.30221.2.3.4 USAGE
 userApplications )
objectClasses: ( jsonObjectClass-OID NAME 'jsonObjectClass' AUXILIARY MAY
 jsonAttr1 )
```

Note: The EQUALITY matching rule should always be specified as `jsonObjectExactMatch` in the schema definition. Using the `jsonObjectFilterExtensibleMatch` is not valid in this case.

Two matching rules are provided to filter the JSON object syntax: `jsonObjectExactMatch` and `jsonObjectFilterExtensibleMatch`.

The `jsonObjectExactMatch` equality matching rule is used in evaluating equality filters in search operations, and for matching performed against JSON object attributes for add, compare, and modify operations. It determines whether two values are logically-equivalent JSON objects. The field names used in both objects must match exactly (although fields may appear in different orders). The values of each field must have the same data types. String values are case-insensitive. The order of elements in arrays is considered significant. Substring or approximate matching is not supported.

The `jsonObjectFilterExtensibleMatch` matching rule can perform more powerful matching against JSON objects. The assertion values for these extensible matching filters should be JSON objects that express the constraints for the matching. These JSON object filters are described in detail in the Javadoc documentation for the LDAP SDK for Java. Although the LDAP SDK can facilitate searches with this matching rule, these searches can be issued through any LDAP client API that supports extensible matching.

The following are example searches using the `jsonObjectFilterExtensibleMatch` rule with available filter types.

Equals field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "equals", "field" : ["stuff", "onetype", "name"], "value" : "John Doe" })'
```

Contains field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "containsField", "field" : "age", "expectedType" : "number" })'
```

Greater than field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "greaterThan", "field" : "age", "value" : 26, "allowEquals" : true})'
```

Configuring JSON Attribute Constraints

The PingDirectory Server can define a number of constraints for the fields included in JSON objects stored in values of a specified attribute type. Constraints that can be placed on a JSON field include:

- Requiring values of the field to have a specified data type.
- Indicating whether the field is required or optional.
- Indicating whether the field can have multiple values in an array. If a field is permitted to have array values, restrictions can also be placed on the number of elements that can be present in the array.
- Indicating whether the field can have a value that is the null primitive as an alternative to values of the indicated data type.
- Restricting values of string fields to a predefined set of values, that match a given regular expression, or a specified length.
- Restricting values of numeric fields with upper and lower bounds.

Any existing data that doesn't match newly-defined JSON constraints can still be decoded and managed by the server. Only new entries are subject to the new constraints. Attempts to alter existing entries with non-compliant JSON objects may require fixing those objects to make them conform to the new constraints.

The two global configuration properties that define schema constraints for JSON objects are `create-json-attribute-constraints` and `create-json-field-constraints` in `dsconfig`. In `dsconfig` interactive under advanced settings, the menu options are JSON Attribute Constraints and JSON Field Constraints. Configuration properties for each include:

- **attribute-type**. The name or object identifier of the attribute type with which the definition is associated. This attribute type must have the JSON object syntax. This property will be the naming attribute for the configuration entry.
- **allow-unnamed-fields**. A boolean value that indicates whether JSON objects, used as the values of attributes of the associated type, can include fields that are not referenced in the `attribute-value-constraints` object. If this is `false`, JSON objects will only be permitted to have the defined fields. If this is `true` (which is the default behavior), JSON objects are permitted to have fields that are not referenced, and no constraints are imposed on those fields.

Unless a schema definition is configured with `allow-unnamed-fields` set to `false`, it is only necessary to include information about fields whose values should be indexed or tokenized. However, it may be desirable to define other fields that are expected in order to ensure that clients will not be permitted to store invalid values. See the Working with JSON Indexes for information about indexing JSON attributes.

As with standard LDAP schema, JSON schema constraints are enforced for any changes made after the constraints are defined. If there are already JSON values in the data before a JSON schema is defined for that attribute type (or before changes are made), values that already exist may violate those constraints. JSON schema constraints will also be enforced for data provided in an LDIF import, so that entries containing JSON objects that violate these constraints will be rejected.

Table 42: JSON Field Constraints

Property	Description
`field`	Specifies the path to the target field as a string, with periods to separate levels of hierarchy. If any field name in the hierarchy

Property	Description
	itself includes a period, that period should be escaped with a backslash.
`value-type`	Specifies the expected data type for the target field. Values can include: • `any`. The target field can have any value. • `boolean`. The target field must have a value of true or false. • `integer`. The target field must have a number that can be exactly represented as an integer. • `null`. The target field must have a value of null. • `number`. The target field must have a value that represents a valid JSON number • `object`. The target field must have a value that represents a valid JSON object. The `allowed-fields` array may contain additional elements that define constraints for the fields that may be present in the object. • `string`. The target field must have a value that represents a valid JSON string.
`is-required`	Specifies whether the target field is required to be present. If it

Property	Description
	is present, its value must be either `true` or `false`. If it is absent, a default of `false` is assumed.
`is-array`	Indicates whether the target field can be an array. If the value can be an array, all of the elements of the array must be of the type specified in the `value-type` field. It can be present with a value that is one of the single strings listed below. If it is absent, a value of `prohibited` is assumed. Values include: • `required`. The target field must be an array with zero or more values of the specified type, and must not be a single value of the specified type. • `optional`. The target field may be an array with zero or more values of the specified type, or it may be a single value of the specified type. • `prohibited`. The target field must not be an array and may only be a single value of the specified type.
`allow-null-value`	Specifies whether the target field can have a value of `null` as an alternative to the specified `value-type`. If present, its

Property	Description
	value must be either true or false. If it is absent, a default of false is assumed. This has no effect if the value-type is null.
allow-empty-object	If the value of the target field is a JSON object (or an array of JSON objects), specifies whether empty objects are permitted. If present, the value must be either true (the object may have zero or more fields) or false (the object may have one or more fields). If it is absent, a default of false is assumed.
index-values	Specifies whether values of the target field should be indexed in backends. If present, the value must be either true (the field should be indexed) or false (the field should not be indexed). If it is absent, a default of false is assumed. If true, the value-type must be boolean, integer, null, or string.
index-entry-limit	Specifies the maximum number of entries in which a particular value may appear before the entry ID list for that value is no longer maintained. If present, the value must be an integer greater than or equal to one. If it is absent, the server

Property	Description
	will use the default index entry limit for the associated backend. This is only applicable if `index-values` is `true`.
`prime-index`	Specifies whether backends should prime the contents of the JSON index database into memory when they are opened. This is ignored if the backend's `prime-all-indexes` property has a value of `true`.
`cache-mode`	Specifies the cache mode to use for the contents of the JSON index database. If the value is not specified, the backend's default cache mode will be used. If a cache mode of `cache-keys-only` is configured, priming will only load the internal nodes into memory for the index. If `no-caching` is configured, no priming is performed for the index.
`tokenize-values`	Specifies whether values of the target field should be tokenized in backends. If present, the value is either `true` (field values should be tokenized) or `false`. If it is absent, a value of `false` is assumed. If `true`, the `value-type` must be `string`.
`allowed-values`	Specifies an explicit set of values that

Property	Description
	the target field is permitted to have. If present, the value must be an array of strings. Any attempt to use a value not in this array for the associated field is rejected. This can only be used with a `value-type` of `string`. If not present, any value can be used as long as it satisfies all other defined constraints.
`allowed-value-regular-expression`	Specifies a regular expression that the target field will be required to match. If present, the value must be a single string or an array of strings representing valid regular expressions (which may require escaping to represent in JSON). Any attempt to use a value that does not match one of the provided regular expressions will be rejected. This can only be used with a `value-type` of `string`. If not present, values are not required to match any regular expression.
`minimum-numeric-value`	Specifies the lower bound for values of the target field. If present, the value must be a single number (it can be an integer). Any attempt to use a value that is less than this number is rejected. This can only be used with a `value-`

Property	Description
	type of integer or number. If not present, no minimum numeric value applies.
maximum-numeric-value	Specifies the upper bound for values of the target field. If present, the value must be a single number (it can be an integer). Any attempt to use a value that is more than this number is rejected. This can only be used with a value-type of integer or number. If not present, no maximum numeric value applies.
minimum-value-length	Specifies the minimum number of characters allowed for values of the target field. If present, the value must be an integer. Any attempt to use a value with fewer characters than this number is rejected. This can only be used with a value-type of string. If not present, no minimum length applies.
maximum-value-length	Specifies the maximum number of characters allowed for values of the target field. If present, the value must be an integer. Any attempt to use a value with more characters than this number is rejected. This can only be used with a value-type of string. If not

Property	Description
	present, no maximum length applies.
`minimum-value-count`	Specifies the minimum number of elements that must be present in an array. If present, the value must be an integer. Any attempt to use an array with fewer elements is rejected. This can only be used with an `is-array` value of `required` or `optional`. If not present, no minimum array value count applies.
`maximum-value-count`	Specifies the maximum number of elements that must be present in an array. If present, the value must be an integer. Any attempt to use an array with more elements is rejected. This can only be used with an `is-array` value of `required` or `optional`. If not present, no maximum array value count applies.

Note: When writing JSON objects in a local DB backend, field names and JSON primitive values of `null`, `true`, and `false` are always tokenized. Integers are either tokenized or compacted using their two's complement representation (other numbers are stored using string representations). Array and object sizes are compacted, and their contents are compacted based on their data types. String values are tokenized that match a recognizable format, including:

- Dates and times in common generalized time and ISO 8601 formats.
- UUIDs in which the alphabetic characters are either all uppercase or all lowercase.
- Strings of at least 12 bytes that are a valid base64 encoding.
- Strings of at least 6 bytes that are a valid hexadecimal encoding, in which the alphabetic characters are either all uppercase or all lowercase.

To Add Constraints to JSON Attributes

- Use the `dsconfig` tool (interactive or command line) to create and configure JSON attribute constraints. The following is a sample command:

```
$ bin/dsconfig create-json-attribute-constraints \
  --attribute-type appjson \
  --set enabled:true \
  --set allow-unnamed-fields:false
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.verified \
  --set value-type:boolean \
  --set is-required:true \
  --set index-values:true \
  --set tokenize-values:false \
  --set allow-null-value:true
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.type \
  --set value-type:string \
  --set is-required:true \
  --set index-values:true \
  --set tokenize-values:true \
  --set allowed-value:home \
  --set allowed-value:other \
  --set allowed-value:work \
  --set allow-null-value:false \
  --set minimum-value-length:1
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.value \
  --set value-type:string \
  --set is-required:true \
  --set index-values:true \
  --set tokenize-values:true \
  --set prime-index:true \
  --set allow-null-value:true \
  --set maximum-value-length:256 \
  --set minimum-value-length:1 \
  --set allowed-value-regular-expression:[-_\+\.\w\d]+@\w+\.\w{2,5}
```

Chapter 18

Managing Password Policies

Topics:

The PingDirectory Server provides a flexible password policy system to assign, manage, or remove password policies for root and non-root users. The password policy contains configurable properties for password expiration, failed login attempts, account lockout and other aspects of password and account maintenance on the Directory Server. The Directory Server also provides a global configuration option and a per-user privilege feature that disables parts of the password policy evaluation for production environments that do not require a password policy.

This chapter presents the following topics:

Viewing Password Policies

Password policies enforce a set of rules that ensure that access to data is not compromised through negligent password practices. The PingDirectory Server provides mechanisms to create and maintain password policies that determine whether passwords should expire, whether users are allowed to modify their own passwords, or whether too many failed authentication attempts should result in an account lockout. Many other options are available to fully configure a password policy for your PingData Platform system.

The Directory Server provides three out-of-the-box password policies that can be applied to your entries or as templates for configuring customized policies. The Default Password Policy is automatically applied to all users although it is possible to use an alternate password policy on a per-user basis. The Root Password Policy is enforced for the default root user, which uses a stronger password storage scheme (PBKDF2 rather than the salted 256-bit SHA-2 scheme) and also requires that a root user provide his or her current password to select a new password.

The Secure Password Policy provides a more secure option than the default policy that makes use of a number of features, including password expiration, account lockout, last login time and last login IP address tracking, password history, and a number of password validators.

CAUTION:

Using the Secure Password policy as-is may notably increase write load in the server by requiring updates to password policy state attributes in user entries and/or requiring users to change passwords more frequently. In environments in which write throughput is a concern (including environments spread across multiple data centers requiring replication over a WAN), it may be useful to consider whether the policy should be updated to reduce the number of entry updates that may be required.

To View Password Policies

- You can view the list of password policies configured on the Directory Server using the `dsconfig` tool, in either interactive or non-interactive mode, or the Administrative Console. The following example demonstrates the process for obtaining a list of defined password policies in non-interactive mode:

```
$ bin/dsconfig list-password-policies
```

```
Password Policy      : Type : password-attribute : default-password-storage-
scheme
------------------------:---------:--------------------:------------------------------
Default Password Policy : generic : userpassword    : Salted 256-bit SHA-2
Root Password Policy : generic : userpassword    : PBKDF2
Secure Password Policy  : generic : userpassword       : PBKDF2
```

To View a Specific Password Policy

- Use `dsconfig` or the Administrative Console to view a specific password policy. In this example, view the Default Password Policy that applies to all uses for which no specific policy has been configured.

```
$ bin/dsconfig get-password-policy-prop \
  --policy-name "Default Password Policy"
```

```
Property                                  : Value(s)
------------------------------------------:--------------------------
description                               : -
password-attribute                        : userpassword
default-password-storage-scheme           : Salted SHA-1
deprecated-password-storage-scheme        : -
password-validator                        : -
account-status-notification-handler       : -
allow-user-password-changes               : true
password-change-requires-current-password : false
```

```
force-change-on-add                        : false
force-change-on-reset                      : false
password-generator                         : Random Password Generator
require-secure-authentication              : false
require-secure-password-changes            : false
min-password-age                           : 0s
max-password-age                           : 0s
max-password-reset-age                     : 0s
password-expiration-warning-interval       : 5d
expire-passwords-without-warning           : false
allow-expired-password-changes             : false
grace-login-count                          : 0s
lockout-failure-count                      : 0s
lockout-duration                           : 0s
lockout-failure-expiration-interval        : 0s
require-change-by-time                     : -
last-login-time-attribute                  : ds-pwp-last-login-time
last-login-time-format                     : -
previous-last-login-time-format            : -
idle-lockout-interval                      : 0s
password-history-count                     : 0s
password-history-duration                  : 0s
```

About the Password Policy Properties

The Directory Server provides a number of configurable properties that can be used to control password policy behavior.

Note: To view a description of each of the password policy properties, see the *Ping Identity Directory Server Configuration Reference* that is bundled with the PingDirectory Server.

Some of the most notable properties include:

- **allow-user-password-changes**. Specifies whether users can change their own passwords. If a user attempts to change his/her own password, then the server will consult this property for the user’s password policy, and will also ensure that the access control handler allows the user to modify the configured password attribute.
- **default-password-storage-scheme**. Specifies the names of the password storage schemes that are used to encode clear-text passwords for this password policy.
- **enable-debug**. When enabled, is used to debug password policy interaction. This property should be used in addition to the server's debug framework with a relevant debug target.
- **force-change-on-add**. Specifies whether users are required to change their passwords upon first authenticating to the Directory Server after their account has been created.
- **force-change-on-reset**. Specifies whether users are required to change their passwords after they have been reset by an administrator. An administrator is a user who has the `password-reset` privilege and the appropriate access control instruction to allow modification of other users’ passwords.
- **idle-lockout-interval**. Specifies the maximum length of time that an account may remain idle (the associated user does not authenticate to the server) before that user is locked out. For accounts that do not have a last login time value, the password changed time or the account creation time will be used. If that information is not available, then the user will not be allowed to authenticate. It is strongly recommended that the server be allowed to run for a period of time with last login time tracking enabled (i.e., values for both `last-login-time-attribute` and `last-login-time-format` properties) to ensure that users have a last login time before enabling idle account lockout.
- **lockout-duration**. Specifies the length of time that an account is locked after too many authentication failures. The value of this attribute is an integer followed by a unit of seconds, minutes, hours, days, or weeks. A value of 0 seconds indicates that the account must remain locked until an administrator resets the password.
- **lockout-failure-count**. Specifies the maximum number of times that a user may be allowed to attempt to bind with the wrong password before that user’s account becomes locked either temporarily (in which case the account will automatically be unlocked after a configurable length of time) or permanently (in which case an administrator

must reset the user's password before the account may be used again). For example, if the value is set to 3, the user is locked out after three failed attempts, even if a fourth attempt is made with the correct password.

- **max-password-age**. Specifies the maximum length of time that a user can continue to use the same password before he or she is required to choose a new password. The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w). A minimum length of time can also be specified before the user is allowed to change the password.
- **password-change-requires-current-password**. Specifies whether users must include their current password when changing their password. This applies for both password changes made with the password modify extended operation as well as simple modify operations targeting the password attribute. In the latter case, if the current password is required then the password modification must remove the current value and add the desired new value (providing both the current and new passwords in the clear rather than using encoded representations).
- **password-expiration-warning-interval**. Specifies the length of time before a user's password expires that he or she receives notification about the upcoming expiration (either through the password policy or password expiring response controls). The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w).
- **password-retirement-behavior**. Specifies the behavior of a password that is allowed a retirement period before becoming invalid. This setting may be used by application service accounts that require a transition period while updating passwords. This is disabled by default.
- **password-validator**. Specifies the names of the password validators that are used with the associated password storage scheme. The password validators are invoked when a user attempts to provide a new password, to determine whether the new password is acceptable.
- **require-secure-authentication**. Indicates whether users with the associated password policy are required to authenticate in a secure manner. This might mean either using a secure communication channel between the client and the server, or using a SASL mechanism that does not expose the credentials.
- **require-secure-password-changes**. Indicates whether users with the associated password policy are required to change their password in a secure manner that does not expose the credentials.

Modifying an Existing Password Policy

You can modify an existing password policy to suit your company's requirements.

To Modify an Existing Password Policy

- Use dsconfig (in interactive or non-interactive mode) or the Administrative Console to modify the configuration for any defined password policy. The following example sets some of the properties presented in the previous section for the default password policy using dsconfig in non-interactive mode:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "max-password-age:90 days" \
  --set "password-expiration-warning-interval:14 days" \
  --set "lockout-failure-count:3" \
  --set "password-history-count:6"
```

Creating a New Password Policy

You can create any number of password policies in the Directory Server using either the dsconfig tool (in interactive or non-interactive mode) or the Administrative Console.

To Create a New Password Policy

- Use dsconfig (in interactive or non-interactive mode) or the Administrative Console to create a new password policy. The following example demonstrates the process for creating a new policy using dsconfig in non-interactive mode.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Demo Password Policy" \
  --set "password-attribute:userpassword" \
  --set "default-password-storage-scheme:Salted 256-bit SHA-2" \
  --set "force-change-on-add:true" \
  --set "force-change-on-reset:true" \
  --set "password-expiration-warning-interval:2 weeks" \
  --set "max-password-age:90 days" \
  --set "lockout-duration:24 hours" \
  --set "lockout-failure-count:3" \
  --set "password-change-requires-current-password:true"
```

To Assign a Password Policy to an Individual Account

To indicate that a user should be subject to a particular password policy (rather than automatically inheriting the default policy), include the `ds-pwp-password-policy-dn` operational attribute in that user's entry with a value equal to the DN of the desired password policy for that user. This attribute can be explicitly included in a user's entry, or it can be generated by a virtual attribute, which makes it easy to apply a custom password policy to a set of users based on a flexible set of criteria.

1. Create a file (`assign.ldif`) with the following contents:

```
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password
 Policies,cn=config
```

2. Use `ldapmodify` to apply the modification to the user's entry.

```
$ bin/ldapmodify --filename assign.ldif
```

To Assign a Password Policy Using a Virtual Attribute

It is possible to automatically assign a custom password policy for a set of users using a virtual attribute. The virtual attribute can be configured so that it can use a range of criteria for selecting the entries for which the virtual attribute should appear.

1. Create an LDIF file, which may be used to add a group to the server.

```
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: Groups

dn: cn=Engineering Managers,ou=groups,dc=example,dc=com
objectClass: groupOfUniqueNames
objectClass: top
cn: Engineering Managers
uniqueMember: uid=user.0,ou=People,dc=example,dc=com ou: groups
```

2. Use `ldapmodify` to add the entries to the server.

```
$ bin/ldapmodify --defaultAdd --filename groups.ldif
```

3. Use `dsconfig` to create a virtual attribute that will add the `ds-pwp-pasword-policy-dn` attribute with a value of `cn=Demo Password Policy,cn=Password Policies,cn=config` to the entries for all users that are members of the `cn=Engineering Managers,ou=Groups,dc=example,dc=com` group.

```
$ bin/dsconfig create-virtual-attribute \
  --name "Eng Mgrs Password Policy" \
  --type user-defined \
  --set "description:Eng Mgrs Grp PWPolicy" \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "value:cn=Demo Password Policy,cn=Password Policies,cn=config" \
  --set "group-dn:cn=Engineering Managers,ou=Groups,dc=example,dc=com
```

4. Use `ldapsearch` to verify that a user in the group contains the assigned password policy DN.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
ds-pwp-password-policy-dn
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password
 Policies,cn=config
```

Deleting a Password Policy

You can delete a password policy using either the `dsconfig` tool (in interactive or non-interactive mode) or the Administrative Console.

To Delete a Password Policy

Custom password policies may be removed using either the `dsconfig` tool (in interactive or non-interactive mode) or the Administrative Console).

- Run the `dsconfig` command with the delete-password-policy subcommand to remove a password policy.

```
$ bin/dsconfig delete-password-policy \
  --policy-name "Demo Password Policy"
```

Modifying a User's Password

There are two primary ways to change user passwords in the Directory Server:

- Perform a modify operation which replaces the value of the password attribute (often `userPassword`). Note that in some configurations, when a user attempts to change his or her own password it may be necessary to perform the modification by removing the password value and adding the desired new value to demonstrate that the user knows the current password value.
- Use the password modify extended operation to change the password. Note that if a user is changing his or her own password, it may be necessary to provide the current password value. The server will allow a new password to be provided (assuming that the new password is acceptable to all configured password validators), or it may automatically generate a new password for the user.

Note that regardless of the mechanism used to change the password, all password values should be provided in clear text rather than pre-encoded, and the user will be required to have sufficient access control rights to update the password attribute in the target user's entry. Further, when one user attempts to change the password for another user, then the requester will be required to have the `password-reset` privilege.

Validating a Password

The requirements that the server will impose for a password change can be displayed to users. The get password quality requirements extended operation can be used to retrieve information about the requirements, which can then be sent to an end user before an attempted password change. These requirements can also be used to enable client-side validation, so that any password problems can be identified before it is sent to the server. The password validation details request control can be included in an add or modify request, or a password modify extended request, to identify which validation requirements were not met by the password provided in the request.

Password validators can be configured with user-friendly messages that describe the password requirements, and the messages that should be returned if a proposed password does not satisfy those requirements. The server will generate these messages if they are not provided in the configuration.

Password validator properties include `validator-requirement-description` and `validator-failure` message. The following is a simple password validator configuration that requires passwords to contain a minimum of five characters, and lists custom validator messages:

```
$ dsconfig create-password-validator \
  --validator-name "Minimum 5 Characters Password Validator" \
  --type length-based --set enabled:true \
  --set "validator-requirement-description:The password must contain
         at least 5 characters." \
  --set "validator-failure-message:The password did not contain
         at least 5 characters." \
  --set min-password-length:5
```

After the password validator is created, it should be assigned to a Password Policy to take affect:

```
$ dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Minimum 5 Characters Password Validator"
```

Retiring a Password

An account password can be retired and rotated out of service, instead of immediately invalidated. This enables a new password to be assigned to an account while keeping the original password valid for a period of time to enable a transition. This is useful for application service accounts that require uninterrupted authentication with the server.

This behavior is disabled by default, but can be enabled in the password policy configuration by setting the `password-retirement-behavior` and `maximum-retired-password-age` properties.

To manually retire an account password or purge a password that has been retired, use the `ldapmodify` and `ldappasswordmodify` commands with options `-- retireCurrentPassword` and `-- purgeCurrentPassword`. To use these commands on an account, the password policy that governs the account must have the `password-retirement-behavior` enabled.

To Change a User's Password using the Modify Operation

- Use `ldapmodify` to change a user's password by replacing the `userPassword` attribute.

```
$ bin/ldapmodify
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: newpw
```

To Change a User's Password using the Password Modify Extended Operation

- Use `ldappasswordmodify` to request that the Password Modify extended operation be used to modify a user's password.

```
$ bin/ldappasswordmodify --authzID dn:uid=jdoe,ou=People,dc=example,dc=com \
  --newPassword newpw
```

To Use an Automatically-Generated Password

- Use `ldappasswordmodify` to automatically generate a new password for a user.

```
$ bin/ldappasswordmodify --authzID "u:user.1"
```

```
The LDAP password modify operation was successful
Generated Password: fbi27oqy
```

Enabling YubiKey Authentication

Users can be enabled to authenticate with YubiKey devices (available from Yubico), which generate secure one-time passwords, with the UNBOUNDID-YUBIKEY-OTP SASL mechanism. A YubiKey device generates a different password for every authentication attempt, and that one-time password is sent to a validation service to ensure that it is genuine and has not been used in an earlier authentication attempt. Although it is possible to use this one-time password as the only proof of identity, it is typically combined with a static password as a form of two-factor authentication.

YubiKey authentication requires server configuration and the addition of this capability to a user entry. Configuration of a client ID and API key to use when communicating with the validation service is also required. The API key is a shared secret between the YubiKey validation service and the client that is interacting with it, and is used when generating digital signatures so that both the server and the YubiKey validation service can ensure that the peer server is genuine.

All server and user entry configuration details are available in the *Ping Identity Security Guide*.

Enabling Social Login

Authentication involving credentials that do not reside in, or cannot be forwarded to or validated by the Directory Server (such as social login through Facebook, Google, or Twitter) can be enabled with the the UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION SASL mechanism. The bind request will not include any credentials, and authentication with this mechanism will not actually change the state of the underlying client connection. The server will behave as if the bind request included the retain identity request control, regardless of whether or not that control was included.

Bind requests using this mechanism can include any request controls that are permitted with other bind requests. If the externally-processed authentication is successful, the client can include the get password policy state issues request control in the bind request to obtain information about any password policy state issues that can cause the Directory Server authentication attempt to fail. The password policy request control can also be included to obtain certain password policy state warnings and errors, or to look for the password expired/password expiring controls in the bind response.

All server and user entry configuration details are available in the *PingData Security Guide*.

Managing User Accounts

The Directory Server provides a user management utility, the `manage-account` tool, that provides a means to quickly view and manipulate a number of password and account policy properties for a user or group of users.

Note: A disabled account status (for example, `account-is-disabled: true`) is different from an *account lockout* due to password policy. Unlocking a user account can be done with the `manage-account` tool. A disabled account requires the administrator to enable the account; password resets are not involved.

Ping also hosts the Self Service Account Manager project at https://github.com/pingidentity/ssam, which is a customizable web application allowing users to perform their own account registration, profile updates, and password changes. The project is for testing and development purposes, and is not a supported Ping application.

To Return the Password Policy State Information

- Use `manage-account` to get information about the account's password policy.

```
$ bin/manage-account get-all \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Password Policy DN: cn=Demo Password Policy,cn=Password Policies,cn=config
Account Is Disabled: false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time: 19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration: 1209600
Seconds Until Password Expiration Warning: 0
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count: 3
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset: false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count: 0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

To Determine Whether an Account is Disabled

- Use `manage-account` to determine whether a user's account has been disabled.

```
$ bin/manage-account get-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: true
```

To Disable an Account

- Use `manage-account` to disable a user's account.

```
$ bin/manage-account set-account-is-disabled \
  --operationValue true --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: true
```

To Enable a Disabled Account

- Use `manage-account` to enable a user's account.

```
$ bin/manage-account clear-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: false
```

To Assign the Manage-Account Access Privileges to Non-Root Users

Non-root users (e.g., `uid=admin`) with admin right privileges require access control permission to interact with certain password policy operational attributes when using the `manage-account` tool.

For example, the presence of the `ds-pwp-account-disabled` operational attribute in an entry determines that the entry is disabled. If the non-root admin user does not have the access privilege to read or interact with the `ds-pwp-account-disabled` operational attribute, the `manage-account` tool may report that the account is active. An account is considered active if the `ds-pwp-account-disabled` operational attribute does not exist in the entry or if the admin user does not have permission to see it.

Use the following procedure to give access rights to the non-root admin user.

1. Create a non-root user admin account, such as `uid=admin,dc=example,dc=com`. Grant the `password-reset` privilege to the account. See step 1 and 6 in the Configuring Administrators section for more information.
2. Run the `manage-account` tool to view the account status for an account.

```
$ bin/manage-account get-all \
  --targetDN uid=user.0,ou=People,dc=example,dc=com
```

```
Password Policy DN:  cn=Default Password Policy,cn=Password
 Policies,cn=config
Account Is Disabled:  false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time:  19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:
Seconds Until Password Expiration Warning:
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset:  false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count:  0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

3. Grant access control privileges to an account. The following allows access to manage accounts to a helpdesk user. Depending on the configuration requirements, this user may also need the `permit-get-password-policy-state-issues` and `password-reset` privileges.

```
dn: dc=example,dc=com
 changetype: modify
 add: aci
 aci: (targetattr="userPassword||ds-pwp-last-login-time||ds-pwp-password-
changed-by-required-time||ds-pwp-reset-time||ds-pwp-warned-time||
 ds-pwp-account-disabled||ds-pwp-account-expiration-time||ds-pwp-password-
policy-dn||ds-pwp-auth-failure||ds-pwp-last-login-ip-address||
 ds-pwp-retired-password||ds-pwp-account-activation-time||pwdReset||
pwdChangedTime||pwdAccountLockedTime")
 (version 3.0; acl "Grant full access to PWP related attributes to
 helpdesk"; allow (all) userdn="ldap:///uid=helpdesk,dc=example,dc=com";)
```

4. Run the `manage-account` tool to disable an account. The following command sets the `account-is-disabled` property to true for the `uid=user.0,dc=example,dc=com`.

```
$ bin/manage-account set-account-is-disabled \
  --targetDN uid=user.0,ou=People,dc=example,dc=com \
  --operationValue true
```

```
Account Is Disabled:  true
```

5. Run the `ldapsearch` tool to view the presence of the `ds-pwp-account-disabled` operational attribute in the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" "+"
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-account-disabled: true
```

Disabling Password Policy Evaluation

The Directory Server provides a global configuration property (disable-password-policy-evaluation) that can be used to disable most password policy evaluation processing. This provides a convenience for those production environments that do not require password policy support. If the disable-password-policy property is set to true, passwords will still be encoded and evaluated, but only account expiration and account disabling will be in effect. All other password policy properties, such as password expiration, lockout, and force change on add or reset, are ignored.

The server also supports the use of a bypass-pw-policy privilege, which can be used to skip password policy evaluation for operations on a per-user basis. If a user has this privilege, then they will be allowed to perform operations on user entries that would normally be rejected by the password policy associated with the target entry. Note that this privilege will not have any effect for bind operations.

To Globally Disable Password Policy Evaluation

- Use dsconfig to set the disable-password-policy-evaluation property globally for the Directory Server.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set "disable-password-policy-evaluation:false"
```

To Exempt a User from Password Policy Evaluation

- Use ldapmodify to add the bypass-pw-policy privilege to a user entry.

```
$ bin/ldapmodify
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: bypass-pw-policy
```

Managing Password Validators

A password validator is a password policy component that is used to determine if a new password is acceptable. A password policy can be configured with any number of password validators. If a password policy is configured with multiple password validators, then all of them must consider a proposed new password acceptable before it will be allowed.

Password Validators

The PingDirectory Server offers a number of types of password validators, including those listed in the following table. It is also possible to use the Server SDK to create custom password validators with whatever constraints are necessary for your environment.

Table 43: Password Validators

Password Validators	Description
Attribute Value	Ensures that the proposed password does not match the value of another attribute in the user’s entry. The validator can be configured to look in all attributes or in a subset of attributes. It can perform forward and reverse mapping, and it can also reject values which are substrings of another attribute.
Character Set	Ensures that the proposed password contains a sufficient number of characters from one or more user-defined character sets. For example, the validator can

Password Validators	Description
	ensure that passwords must have at least one lowercase letter, one uppercase letter, one digit, and one symbol.
Commonly-Used Passwords Dictionary	Ensures that the proposed password is not one of 10,000 commonly used passwords. These are words that are common for attackers to use when trying to access user accounts. The Commonly-Used Passwords validator is invoked by the Secure Password Policy by default. The word list is located in `<server-root>/config/commonly-used-passwords.txt`, and can be used to create a custom validator, but should not be modified.
Dictionary	Ensures that the proposed password is not present in a specified dictionary file, optionally also testing the password with all characters in reverse order. A large dictionary file is provided with the server, but the administrator can supply an alternate dictionary. In this case, then the dictionary must be a plain-text file with one word per line.
Haystack Password Validator	Ensures that the proposed password is secure based on a combination of its length and the types of characters that it contains. For example, a longer password containing only lowercase letters may be stronger than a shorter password containing a mix of uppercase and lowercase letters, numbers, and symbols. This is based on the Gibson Research Corporation Password Haystacks concept.
Length-Based Password Validator	Ensures that the number of characters in the proposed new password is within an acceptable range. Both a maximum and minimum number of characters may be specified.
Regular Expression Validator	Ensures that a proposed password either matches or does not match a given regular expression.
Repeated Characters	Ensures that a proposed password does not contain a substring in which the same character is repeated more than a specified number of times (for example, "aaaaa" or "aaabbb"). The validator can be configured to operated in a case-sensitive or case-insensitive manner, and you can also define custom sets of equivalent characters (for example, you could define all digits as equivalent, so the proposed password could not contain more than a specified number of consecutive digits.
Similarity-Based Password Validator	Ensures that the proposed new password is not too similar to the current password, using the Levenstein Distance algorithm, which calculates the number of characters that need to be inserted, removed, or replaced to transform one string into another. Note that for this password validator to be effective, it is necessary to have access to the user's current password. Therefore, if this password validator is to be enabled, the `password-change-requires-current-password` attribute in the password policy configuration must also be set to true.
Unique Characters	Ensures that the proposed password contains at least a specified minimum number of unique characters, optionally using case-insensitive validation.

Configuring Password Validators

You can use the `dsconfig` configuration tool or the Administrative Console to configure or modify any password validators. Once you have defined your password validators, you can add them to an existing password policy. The following example procedures show the dsconfig non-interactive commands necessary to carry out such tasks. If you use `dsconfig` in interactive command-line mode, you can access the Password Validator menu in the Basic Objects menu. For more details on the password validator properties, see the *PingDirectory Server Configuration Reference*.

To View the List of Defined Password Validators

- Use dsconfig to view the set of password validators defined in the Directory Server.

```
$ bin/dsconfig --no-prompt list-password-validators
```

To Configure the Attribute Value Password Validator

1. Use dsconfig to edit the existing default configuration for the Attribute Value Password Validator. For example, the following change configures the validator to only examine a specified set of attributes..

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Attribute Value" \
  --set match-attribute:cn \
  --set match-attribute:sn \
  --set match-attribute:telephonenumber \
  --set match-attribute:uid
```

2. Update an existing password policy to use the Attribute Value Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Attribute Value"
```

3. Test the Attribute Value Password Validator by submitting a password that is identical to one of the configured attributes (cn, sn, telephonenumber, uid).

```
$ bin/ldappasswordmodify --authzID "uid=user.0,ou=People,dc=example,dc=com"
 \
  --newPassword user.0
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks
 defined in the
server: The provided password was found in another attribute in the user
 entry
```

To Configure the Character Set Password Validator

1. Use dsconfig to edit the existing default configuration. In this example, we change the requirement for special characters making them optional in a password, and add a requirement that at least two digits must be included in the password. Thus, in this example, all newly created passwords must have at least one lowercase letter, one uppercase letter, two digits, and optionally any special characters listed.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Character Set" \
  --remove character-set:1:0123456789 \
  --remove "character-set:1:~\!@#\$\%^&*()-_=+[]{}\|;:,.<>/?" \
  --add character-set:2:0123456789 \
  --add "character-set:0:~\!@#\$\%^&*()-_=+[]{}\|;:,.<>/?" \
  --set allow-unclassified-characters:false
```

2. Update an existing password policy to use the Character Set Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Character Set"
```

3. Test the Character Set Password Validator by submitting a password that meets the requirements (one lowercase letter, one uppercase letter, two digits). The following example should reject the given password because it does not pass the Character Set Password Validator.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abab1
```

To Configure the Length-Based Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the required minimum number of characters in a password to five.

```
$ bin/dsconfig create-password-validator \
  --validator-name "Length-Based Password Validator" \
  --set max-password-length:5 --set min-password-length:5
```

2. Update an existing password policy to use the Length-Based Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Length-Based Password Policy"
```

3. Test the Length-Based Password Validator by submitting a password that has fewer than the minimum number of required characters.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abcd
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks
 defined in
the server: The provided password is shorter than the minimum required
 length of
5 characters
```

To Configure the Regular Expression Password Validator

1. Use `dsconfig` to create a Regular Expression password validator. The following password validator checks that the password contains at least one number, one lowercase letter, and one uppercase letter with no restrictions on password length. If the password matches the regular expression, then it will be accepted. When using the following command, remember to include the LDAP/LDAPS connection parameters (host name and port), bind DN, and bind password.

```
$ bin/dsconfig create-password-validator \
  --validator-name "Regular Expression" \
  --type regular-expression --set enabled:true \
  --set "match-pattern:^\\w*(?=\\w*\\d)(?=\\w*[a-z])(?=\\w*[A-Z])\\w*\$" \
  --set match-behavior:require-match
```

2. Update an existing password policy to use the Regular Expression validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Regular Expression"
```

3. Test the Regular Expression Validator by submitting a password that meets the requirements (contains one number, one lowercase letter, and one uppercase letter), then run it again with a password that does not meet these requirements.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaA1
```

```
The LDAP password modify operation was successful
```

4. Try another password. The following password should fail, because no uppercase letter is present.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaa1
```

```
Error Message: The provided new password failed the validation checks
```

```
defined in the server: The provided password is not acceptable because it
 does
not match regular expression pattern '^\w*(?=\w*\d)(?=\w*[a-z])(?=\w*[A-
Z])\w*$'
```

To Configure the Repeated Character Password Validator

1. Use `dsconfig` to edit the existing default configuration.

   - In this example, we set the maximum consecutive length of any character to 3. For example, the following validator rejects any passwords, such as "baaaa1" or "4eeeeb", etc.

     ```
     $ bin/dsconfig set-password-validator-prop \
       --validator-name "Repeated Characters" \
       --set max-consecutive-length:3
     ```
   - Or, you can configure the validator to reject any character from a pre-defined character set that appears more than the specified number of times in a row (2). You can also specify more than one character set. For example, the following validator defines two characters sets: [abc] and [123]. It rejects any passwords with more than two consecutive characters from a character set. Thus, "aaa", "bbb", "ccc", "abc", or "123" and so on fails, but "12a3" is okay.

     ```
     $ bin/dsconfig set-password-validator-prop \
       --validator-name "Repeated Characters" \
       --set character-set:123 --set character-set:abc
     ```
2. Update an existing password policy to use the Repeated Character Password Validator.

   ```
   $ bin/dsconfig --no-prompt set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Repeated Characters"
   ```
3. Test the Repeated Character Validator by submitting a password that has more than the maximum allowable length of consecutive characters.

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" \
     --newPassword baaa1
   ```

   ```
   The LDAP password modify operation failed with result code 53
   Error Message: The provided new password failed the validation checks
    defined
   in the server: The provided password contained too many instances of the
    same
   character appearing consecutively. The maximum number of times the same
   character may appear consecutively in a password is 2
   ```

To Configure the Similarity-Based Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the minimum number of differences to 2.

   ```
   $ bin/dsconfig set-password-validator-prop \
     --validator-name "Similarity-Based Password Validator" \
     --set min-password-difference:2
   ```
2. Update an existing password policy to use the Similarity-Based Password Validator. The `password-change-requires-current-password` property must be set to TRUE, so that the password policy will ensure that the user's current password is available when that user is choosing a new password.

   ```
   $ bin/dsconfig set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Similarity-Based Password Validator" \
     --set password-change-requires-current-password:true
   ```

3. Test the Similarity-Based Password Validator by submitting a password that has fewer than the minimum number of changes (e.g., 2). The `ldappasswordmodify` command requires the --currentPassword option when testing the Similarity-Based Password Validator.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --currentPassword abcde --newPassword abcdd
```

```
The LDAP password modify operation failed with result code 49
```

To Configure the Unique Characters Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the minimum number of unique characters that a password is allowed to contain to 3.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Similarity-Based" --set min-unique-characters:3
```

2. Update an existing password policy to use the Unique Characters Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Unique Characters"
```

3. Test the Unique Characters Password Validator by submitting a password that has fewer than the minimum number of unique characters (e.g., 3).

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword aaaaa
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks
 defined
in the server: The provided password does not contain enough unique
 characters.
The minimum number of unique characters that may appear in a user password
 is 3
```

Chapter 19

Managing Replication

Topics:

The PingDirectory Server provides a robust replication system to ensure high availability and fast failover in production environments. Write requests can be handled by every server in the topology with the replication component performing immediate synchronization with other members. The replicated server environment ensures that LDAP clients can seamlessly fail over to another server instance.

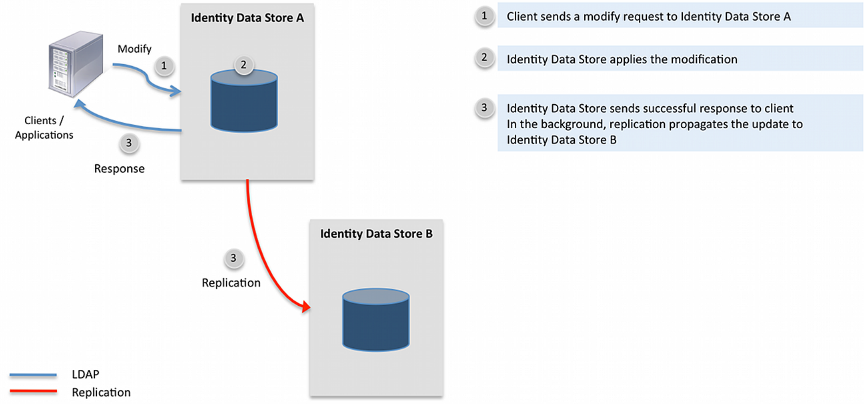
This chapter presents the architectural overview of replication, detailed configuration steps, ways to monitor replication as well as troubleshooting steps:

Overview of Replication

Replication is a data synchronization mechanism that ensures that updates made to a database are automatically replayed to other servers. Replication improves data availability when unforeseen or planned outages occur, and improves search performance by allowing client requests to be distributed across multiple servers.

By default, all Directory Servers participating in replication are writable, so that LDAP clients can perform updates at any of these Directory Server instances. These updates will be propagated to the other servers automatically in the background and applied in the same order as the updates were entered. The replication process flow is designed to immediately propagate changes to the other replicas in the topology with little or no latency.

The following figure demonstrates the basic flow of replication.



1 Client sends a modify request to Identity Data Store A

2 Identity Data Store applies the modification

3 Identity Data Store sends successful response to client
In the background, replication propagates the update to Identity Data Store B

Figure 10: Replication Process Flow

The benefits of replication can be summarized as follows:

- **High-Availability**. Because the data is fully replicated on all other servers in the topology, replication allows participating servers to process all types of client requests. This mitigates any availability issues when a particular server is down due to a planned maintenance or unplanned outage. For those servers that are temporarily unavailable, they will receive updates when they become available again.
- **Improved Search Performance**. Search requests may be directed to any Directory Server participating in replication, which improves search performance over systems that only access single servers. Note, however, that replication does not improve write throughput since updates need to be applied at all servers.
- **WAN Friendly Data Synchronization**. The built-in compression feature in the replication protocol allows efficient propagation of updates over WAN links.

Replication Versus Synchronization

Replication is not a general purpose synchronization facility as it creates replicas with exact copies of the replicated data. Synchronization, on the other hand, can transform data between two different Directory Information Tree (DIT) structures, map attribute types, synchronize subsets of branches and specific object classes. The differences between replication and synchronization are illustrated as follows:

- **Replication cannot Synchronize between Different DIT Structures**. The DN of replicated entries must be the same on all servers. In some situations, it may be desirable to replicate entries with the help of DN mapping that are under different base DNs, but represent the same data, for example `uid=john.doe,ou=people,o=corp` on one server may represent the same user as `uid=john.doe,ou=people,dc=example,dc=com`. This is not supported by replication. Synchronization fully supports this feature.
- **Replication cannot Map Attribute Types or Transform Attribute Values**. In some situations, it may be necessary to map attribute types or transform attribute values when synchronizing data from one server to another. Replication does not support either attribute type mappings or attribute value transformations.

- **Replication does Not Support Fractional Replication**. Replication cannot be configured to replicate a subset of the attribute types from the replicated data set. Synchronization fully supports this feature.
- **Replication does Not Support Sparse Replication**. Replication cannot be configured to replicate entries with a particular object class only. Synchronization fully supports this feature.
- **Replication Requires Full Control of Replicated Data**. When two servers participate in replication, both servers implicitly trust each other using public key cryptography and apply all updates received via replication, which is considered an internal operation. While trust between servers is established between two endpoint servers, synchronization does not require full control of the data. Disparate server system endpoints can be synchronized, such as a PingDirectory Server and a RDBMS database endpoint with each fully in control of its own data.

If replication does not meet your data synchronization requirements, consider using the PingData Data Sync Server, which provides the versatility and robust performance required for most production environments.

Replication Terminology

The following replication terms are used throughout this chapter.

Table 44: Replication Terminology

Term	Description
Assured Replication	For applications that require immediate access to replicated data or require data consistency over response time, administrators can configure *assured replication*, where data is guaranteed to replicate *before* the response is returned to the client.
Change Number	A 64-bit number used to consistently order replication updates across the entire topology. It is composed of 3 fields: the timestamp of the update (measured in milliseconds since 00:00:00 UTC, January 1, 1970), the Replica ID, and the Sequence Number.
Conflict	Client updates made at different replicas affecting the same entry may be found in conflict when the updates are replayed at another replica. The Change Number of each update allows most of these conflicts to be resolved automatically. Certain updates, such as adding an entry with different attribute values at two servers simultaneously, result in conflicts that are flagged for required manual resolution.
Eventual Consistency	When not using assured replication, recent updates from LDAP clients are not immediately present at all servers. Out-of-sync data will be eventually synchronized and will be consistent on all servers. The network latency typically controls how long a given update takes to replicate.
Global Administrator	The administrative user with full rights to manage the replication topology. The user is created at the time of replication enable between two non-replicating servers and thereafter copied to newly enabled servers. The replication command-line utility expects the user name of the Global Administrator (by default, admin). The user is stored in `cn=Topology Admin Users,cn=Topology,cn=config`.
Historical Data	Historical Data are records of attribute value changes as a result of an LDAP Modify Operation. Historical Data is stored in the `ds-sync-hist` attribute of the target entry. This information allows replication to resolve conflicts from simultaneous LDAP Modify operations automatically.
Location	The collection of servers that may have similar performance characteristics when accessed from this Directory Server or reside in the same data center or city. A separate location setting may be defined for each data center, such as Austin, London, Chicago, etc. Location settings are used in the selection of the WAN Gateway server.

Term	Description
Modify Conflict	A conflict between two LDAP Modify operations. Conflicts arising from LDAP Modify operations are automatically resolved.
Naming Conflict	Any conflict other than Modify Conflicts. Naming conflicts typically include an operation that changes the DN of the target entry, creates an entry, or deletes an entry at one Replica.
Replica	The component in the Directory Server that handles interaction with a single replication domain, for example, the `dc=example,dc=com` replication domain within the `userRoot` backend.
Replica ID	The unique identifier of a replica that is set automatically in the corresponding Replication Domain configuration entry at each participating server. The replica ID identifies the source of each update in the replication topology.
Replica State	A list of the most recent change numbers of replication updates that have been applied to a replica. There can be at most one change number from each replica in the state. The replica state helps the Replication Server component to determine which updates the Replica has not received yet.
Replication	An automated background process that pushes directory server data changes to all other replicas.
Replication Changelog	A backend maintained by each replication server independently that records updates from each replica. This backend is distinct from the LDAP Changelog and the two should not be confused. The main distinction is as follows: • The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at `<server-root>/db/changelog`. • The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at `<server-root>/changelogDB` and is not accessible by clients and server extensions.
Replication Domain	The data configured for replication as defined by the base DN. Updates to entries at and below the base DN will be replicated.
Replication Replay	When a replica locally applies the update received via a replication server.
Replication Server	A component within the Directory Server process that is responsible for propagating directory server data changes to and from replicas. Each Directory Server instance participating in replication is also running a replication server.
Replication Server ID	The unique identifier of a replication server set automatically in its configuration object at each server in the replication topology. This identifier is used in the connection management code of the replication servers.
Replication Server State	A list of the most recent change numbers of replication updates that have been processed by a replication server. There can be at most one change number from each replica in the topology. The Replication Server State is used to determine which updates need to be sent to other replication servers. Similarly, the replica can use the Replication Server State to identify the set of updates to send to the replication server.
Sequence Number	A field in the change number that indicates the sequence of the client updates at a particular replica. For every update at the replica, the number is incremented. The initial value of the sequence number is 1. The number is stored as a 32-bit integer, but only positive values are used. The sequence number can roll over.
WAN Gateway	The designated replication server that assumes the WAN Gateway role within a collection of co-located replication servers (i.e., servers that are defined with identical location settings). Replication update messages between servers at different locations are routed through WAN gateways. The WAN Gateway role is assigned automatically

Term	Description
	by the protocol based on the server's WAN Gateway Priority setting. If the WAN Gateway server is down for any reason, the server with the next highest WAN Gateway Priority will dynamically assume the WAN Gateway role.
WAN Gateway Priority	The configuration setting that determines which replication server assumes the WAN Gateway role. The replication server with the lowest WAN Gateway Priority value in a location assumes the role of the WAN Gateway. The priority values can be set to 0 (never be a gateway), or any value from 1 (highest priority) to 10 (lowest priority).

Replication Architecture

The major elements of replication in the PingDirectory Server are introduced in this section.

Eventual Consistency

Replication is based on an eventual-consistency model, where updates that are propagated through a connected network of servers will eventually be consistent on all servers over a very short period of time. In a typical update operation, a client application updates an entry or group of entries on the PingDirectory Server with an ADD, DELETE, MODIFY, or MODIFY DN operation. After processing the operation, the Directory Server returns an LDAP response, while concurrently propagating the update to the other servers in the replicated topology. This concurrent processing model allows the client to continue submitting update requests without waiting for a replication completion response from the server. Alternatively, assured replication can be configured for specific write requests that requires local or global consistency, across datacenter locations, before a response is returned to the client. For more information, see *Configuring Assured Replication* on page 346.

To support this processing model, replication never locks the targeted entries at the other Directory Server instances before an update can be made locally. This means that the replicated Directory Servers may have an inconsistent view of the targeted entry for a very short period of time but will catch up as the propagated changes are applied. The eventual-consistency model also allows clients to complete update operations faster, since clients do not have to wait for replication to propagate the change. The rate of update operations remains the same no matter how many Directory Servers participate in replication.

Replicas and Replication Servers

Each Directory Server has an embedded replication server that is responsible for transmitting updates to other replication servers. There is a separate component, called a replica, for each replicating base DN, such as `cn=schema, dc=example,dc=com`. Each replica connects to the embedded replication server running within the Directory Server JVM process.

When a client application updates an entry on the Directory Server, the replica sends an update message to its embedded replication server. The replication server applies the change to the `replicationChangelog` backend repository and then sends an update message to the connected replication server on another directory server. The replication server on other directory servers then passes the change to the appropriate replica, which in turn applies the change to its backend after performing conflict resolution. This standard setup is seen in the figure below.

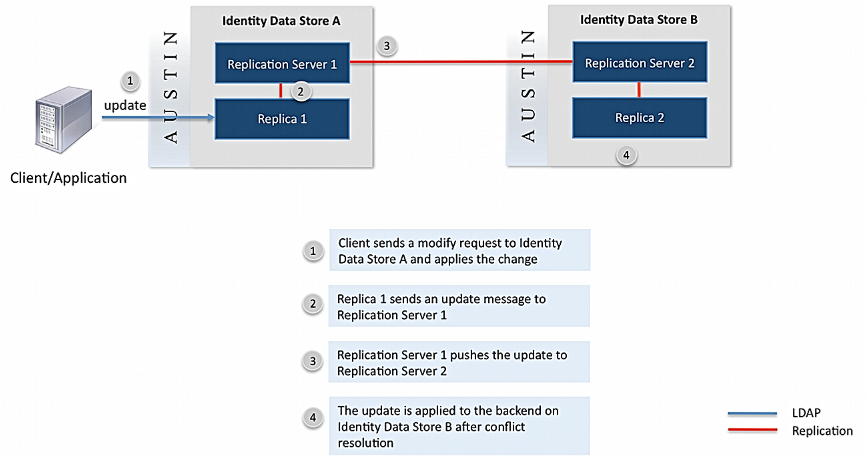

Identity Data Store A
Identity Data Store B
AUSTIN
AUSTIN
Replication Server 1
Replication Server 2
Replica 1
Replica 2
1
update
2
3
4
Client/Application
1 Client sends a modify request to Identity Data Store A and applies the change
2 Replica 1 sends an update message to Replication Server 1
3 Replication Server 1 pushes the update to Replication Server 2
4 The update is applied to the backend on Identity Data Store B after conflict resolution
LDAP
Replication

Figure 11: Replicas and Replication Servers

Authentication and Authorization

The authentication in the Replication Protocol is based on public key cryptography using client certificate authentication via TLS. The certificate used for authentication is stored in the `ads-truststore` backend of the Directory Server. During replication setup, the command-line utility distributes public keys to all directory servers to establish trust between the Directory Servers and to enable client authentication via TLS.

The authorization model of replication is simple: once authenticated, the remote Directory Server is fully authorized to exchange replication messages with the local Directory Server. There is no other access control in place.

Logging

The access log messages in the Directory Server indicate if the update was received via replication and includes the corresponding change number. This allows the administrator to track which Directory Server the update originated from.

Replication Deployment Planning

The following should be considered before deploying in a production environment:

- **Minimizing Replication Conflicts**. Attention should be paid to the origin of client write requests to prevent a conflict. If two different clients attempt to create an entry with the same name, DN, at the same time against different servers, the possibility exists that both client requests will succeed. In this case, a conflict alert will be sent by the server. However, understand the client traffic pattern beforehand will minimize these occurrences. The Directory Server's Assured Replication feature and the PingDirectoryProxy Server can both assist in minimizing conflicts.
- **Replication Purge Delay**. Adjust the default one-day replication purge delay, consistently across all servers, to accommodate automatic catchup of changes when a server is offline for an extended period of time. The replication changes database, stored in `<server-root>/changelogDb`, grows larger as the replication purge delay is increased. A minimum value should be defined.

The rest of this section highlights other topics of consideration.

Location

In multi-site deployments, it is strongly recommended to configure the directory servers with location information using the `dsconfig create-location` command and `dsconfig set-global-configuration-prop` command. The Directory Server cannot determine LAN boundaries automatically, so incorrect location settings can result in undesired WAN communication. By default, replication also compresses all traffic between directory servers in different locations.

We recommend setting up the locations prior to enabling replication. The `dsreplication enable` command prompts for location information if you have forgotten to define the property.

User-Defined LDAP

Directory Servers participating in replication are required to have a uniform user-defined schema. The `dsreplication` command-line utility sets up replication for the schema backend the first time replication is enabled to ensure that future schema changes are propagated to all directory servers.

Disk Space

Replication increases the disk space required for the Directory Server. The Replication Changelog backend keeps changes from all directory servers for 24 hours by default. After this time period, also known as the purge delay, the backend is trimmed automatically.

In addition, within the `userRoot` and other local DN backends, attribute-level changes are recorded for a short period of time in the `ds-sync-hist` attribute of the entry targeted by an LDAP Modify operation. This attribute is used to resolve all conflicts resulting from LDAP Modify operations automatically.

The disk space impact of replication is highly dependent on the rate and size of changes in the replication topology, and the Replication Changelog purge delay.

Memory

Compared to a standalone directory server, replicated directory server instances require slightly more memory. All of the items discussed in the Disk Space section have an impact on the amount of memory the Directory Server is using. The additional replication overhead is typically less than 5%.

Time Synchronization

Even though replication has a built-in mechanism to compensate for the potential clock skew between hosts, it is generally recommended to keep system clocks in sync within the deployment.

Communication Ports

The replication server component in each directory server listens on a TCP/IP port for replication communication (the replication server port). This port, typically 8989, must be accessible from all directory servers participating in replication. The server-to-server communication channel is kept alive using a heartbeat, which occurs every 10 seconds. This traffic will prevent firewalls from closing connections prematurely.

The replication command-line utility (`dsreplication`) requires access to all directory servers participating in replication. This includes the LDAP or LDAPS port of the directory servers.

Keep these communication requirements in mind when configuring firewalls.

Hardware Load Balancers

Replication allows writes to be directed to any directory server in the topology. Distributing write operations in a round-robin fashion, however, may introduce conflicts. In particular, distributing a series of LDAP Add, Delete and Modify DN operations targeting the same DN in quick succession can result in conflicts that require manual intervention. The Assured Replication feature can help prevent conflicts created by the same client application.

If possible, consider using server affinity with the load balancer that either associates a client IP address or a client connection with a single directory server instance at a time. Also, consider using the PingDirectoryProxy Server for load balancing LDAP traffic. The Server Affinity feature in the Directory Proxy Server enables replication-friendly load balancing.

Directory Proxy Server

In addition, to facilitate replication-friendly load balancing, the PingDirectoryProxy Server should be considered in every replication deployment. The Directory Proxy Server can automatically adapt to conditions in backend directory servers using health checks and route traffic accordingly. For example, traffic can be re-routed from directory servers with large backlog of replication updates.

To Display the Server Information for a Replication Deployment

- Run the `dsreplication status` command with the --displayServerTable option.

```
$ bin/dsreplication status --displayServerTable
```

```
          --- Replication Status for dc=example,dc=com: ---
Server                          : Location : Priority (1) : Status
--------------------------------:----------:--------------:----------
austin-01.example.com:8989      : US       : 1 (*)        : Available
austin-02.example.com:8989      : US       : 2            : Available
london-01.example.com:8989      : UK       : 5            : Available
london-02.example.com:8989      : UK       : 5 (*)        : Available
sydney-01.example.com:8989      : AU       : 5 (*)        : Available
sydney-02.example.com:8989      : AU       : 5            : Available

[1] WAN Gateway Priority. WAN gateways are marked with a *. To minimize
WAN utilization, the server with the WAN Gateway role is the only server
in a location that exchanges updates with remote locations.

--- Replication Status for dc=example,dc=com: Enabled ---
Server                          : Location : Entries : Conflict Entries :
 Backlog : Recent Change Rate
--------------------------------:----------:---------:------------------:---------:---
austin-01.example.com:1389      : US       : 3478174 : 0                : 8
      : 333
austin-02.example.com:1389      : US       : 3478174 : 0                : 5
      : 345
london-01.example.com:1389      : UK       : 3478174 : 0                : 0
      : 349
london-02.example.com:1389      : UK       : 3478174 : 0                : 0
      : 5
sydney-01.example.com:1389      : AU       : 3478173 : 0                : 0
      : 350
sydney-02.example.com:1389      : AU       : 3478270 : 0                : 30
      : 332
```

To Display All Status Information for a Replication Deployment

- Run the `dsreplication status` command with the --showAll option. You can also use the --showAll option together with the --displayServerTable option to see the server table information for the replication topology.

Enabling Replication

Enabling replication between multiple Directory Servers means that any change within the replicated base DN is automatically propagated to all other Directory Servers in the topology. Configuration changes are not replicated, but can be through the use of Server Groups. Therefore, each Directory Server should be configured according to the deployment design.

Overview

To interface with the replication topology, the Directory Server provides a command-line utility, `dsreplication`, that must be used to manage and monitor replication.

Replication setup involves the following basic steps:

- **Set up the servers**. This is the basic installation steps to set up a Directory Server instance.

- **Import or restore data to one server**. After setting up the servers, at least one server should have the target data loaded through `import-ldif` or restore.
- **Enable replication between the servers**. Using the `dsreplication` tool, enable replication for each server to be included in the replication topology. The `dsreplication enable` subcommand should be run *N - 1* times for a topology of *N* servers. See Command Line Interface for more information.
- **Initialize data from source server to every server in the topology**. Run the `dsreplication initialize` subcommand for every target server that needs a copy of the data from the source server.
- **Verifying the replication topology**. Administrators can check the replication status after configuring the topology using the `dsreplication` status tool.

Command Line Interface

Replication topologies are configured and maintained using the `dsreplication` command-line utility, which supports interactive and non-interactive modes. If you are running the server for the first time, we recommend using the `dsreplication` tool in interactive mode.

The `dsreplication` tool has the following format including some important subcommands listed in the section The dsreplication Command-Line Utility:

`dsreplication` {subcommand} {connection parameters}

The `dsreplication` tool keeps a history of invocations in the `logs/tools/dsreplication.history` file and keeps a log of up to 10 `dsreplication` sessions in the `logs/tools` directory.

What Happens When You Enable Replication

The `dsreplication` enable subcommand is used to set up replication. The enable subcommand carries out the following functions:

- If it does not already exist, the global administrator user is created. The global administrator user has all the rights and privileges to update replication-related configuration objects. Most `dsreplication` subcommands require the global administrator.
- The server instances are registered in the `cn=topology,cn=config` tree. The registration includes basic host name, port information as well as the public key used during the replication authentication process.

  In case both servers are already participating in replication, the `cn=topology,cn=config` is merged to retain the server information from existing topologies.
- The embedded replication server is enabled. Servers already in replication will see their replication server configuration updated with the information of the new replication server.
- A replication domain is created for the requested base DNs. In case the first base DN is enabled, the replication domains for two additional base DNs are also enabled: `cn=topology,cn=config` and `cn=schema`.
- Initialization for the `cn=topology,cn=config` base DN is executed. This will ensure that `cn=topology,cn=config` is uniform across the replication topology.
- Initialization for the `cn=schema` base DN is executed. This will ensure that a uniform schema is present in the replication topology.
- Initialization must be performed for the enabled base DNs.

Initialization

Replica initialization transfers of a copy of the backend containing the replication domain to a target server. This should be performed after replication is enabled with the `dsreplication initialize` subcommand. There is no impact on the source server during this process.

Note: When enabling or initializing servers, the `--topologyFilePath` option can be used with `dsreplication` to specify a file with a series of hosts and ports available in the topology that can be used as source servers. This option is used in place of specifying host 1, port1. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available for replication. This file is generated with the `manage-topology export` command.

- **dsreplication initialize**. The recommended approach for replica initialization. The `dsreplication initialize` subcommand performs the most efficient copy of data needed to initialize one or more replicas on a target server. Any existing data on the target server replica will be lost and the backend containing the base DN will be taken offline on the target server during the initialization.
- **Binary Copy**. The database copy method involves copying database backup files from the source directory server to one or more target servers. The Directory Server provides tools necessary for backing up and restoring backends. Using `server-root/bin/backup`, create a backup of the backend containing the replicated base DN. The backup files then need to be transferred to the target server(s) and restored individually with `server-root/bin/restore`. There are additional considerations when using database copy as the means to initialize a target replica:
  - If encryption is enabled on the servers, then a database `bin/encryption-settings export` then `bin/encryption-settings import`must be performed on the `encryption-settings` backend.
  - When using database copy to initialize a server which has been offline longer than the replication purge delay, the database copy of the `replicationChanges`, `schema`, and `adminRoot` backends are required.

Replica Generation ID

Each replica has a generation ID, which is an integer that summarizes the replica. It provides replication with a quick and simple means of determining if two replicas contain the same data. If they do contain the same data, they'll have the same generation ID.

When replication is operating correctly, all of the replicas for each replicated base DN will have the same generation ID. The generation ID is stored on each replica as the operational attribute `ds-sync-generation-id`. For example:

```
ldapsearch -b 'dc=example,dc=com' -s base '(&)' ds-sync-generation-id
 dn: dc=example,dc=com
 ds-sync-generation-id: 2058329333
```

When the server starts, or when replication is enabled, the generation ID is computed for each affected replica that does not already have a generation ID stored as `ds-sync-generation-id`. The following is used to calculate the generation ID:

- The total number of entries in the replica. This is referred to as "the count."
- The first 1000 entries in the replica are converted to normalized LDIF, which is referred to as "the LDIF." Normalized LDIF only includes attributes `objectclass`, `sn`, `cn` and `ds-entry-unique-id`, and uses OIDs in place of attribute names.
- The Adler-32 checksum is calculated with the string produced by concatenating the count and the LDIF as input. This Adler-32 checksum is the generation ID.
- The generation ID is stored on the base DN as `ds-sync-generation-id`. This is so that the ID does not need to be computed the next time the replica is loaded.

Deploying a Basic Replication Topology

This section describes how to set up a two-server replication topology. The example uses the LDAP and replication server ports 1389 and 8989 respectively.

Table 45: Replica Ports

Host Name	LDAP Port	Replication Port
server1.example.com	1389	8989
server2.example.com	1389	8989

To Deploy a Basic Replication Deployment

1. Install the first directory server with 2000 sample entries.

```
$ ./setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389
 \ --rootUserPassword pass --sampleData 10000 --no-prompt
```

2. Install the second directory server either on a separate host or the same host as the first, but with a different LDAP port.

```
$ ./setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389
 \ --rootUserPassword pass --no-prompt
```

3. From the first server, run the `bin/dsreplication` command in interactive mode to configure a replication topology:

```
$ bin/dsreplication
```

4. From the Replication Main menu, select the Manage the topology option.

```
>>>> Replication Main Menu

What do you want to do?

    1) Display replication status
    2) Manage the topology (add and remove servers)
    3) Initialize replica data over the network
    4) Initialize replica data manually
    5) Replace existing data on all servers

    q) quit

Enter choice: 2
```

5. From the Manage Replication Topology menu, choose the Enable Replication option.

```
>>>> Manage Replication Topology

Select an operation for more information.

    1)  Enable Replication -- add or re-attach a server to the topology
    2)  Disable Replication -- permanently remove a running replica from the
 topology
    3)  Remove Defunct Server -- permanently remove an unavailable server
 from the
          topology
    4)  Cleanup Server -- remove replication artifacts from an offline,
 local server
          (allowing it to be re-added to a topology)

    b)  back
    q)  quit

Enter choice [b]: 1
```

6. On the Enable Replication menu, read the brief introduction on what will take place during the setup, and then, enter "c" to continue the enable process.
7. Next, enter the LDAP connection parameters for the first of the two replicas that you are configuring. First, enter the host name or IP address of the first server.
8. Next, enter the type of LDAP connection to the first server: 1) LDAP, 2) LDAP with SSL, or 3) LDAP with StartTLS.
9. Type the LDAP listener port for the first replica. If you are a root user, you will see port 389 as the default. Others will see port 1389.

10. Authenticate as a root DN, such as `cn=Directory Manager`. You will be prompted later in the process to set up a global administrator and password. The global administrator is the user ID that manages the replication topology group.
11. Repeat steps 7–10 for the second replica.
12. Next, the `dsreplication` tool checks for the base DN on both servers. In order to enable replication, data must be present on at least one of the servers. For this example, press Enter to select the default base DN, `dc=example,dc=com`.

```
Choose one or more available base DNs for which to enable replication:

    1)  dc=example,dc=com

    c)  cancel

Enter one or more choices separated by commas [1]:
```

Note: If you see the following message:

```
There are no base DNs available to enable replication between the two
 servers.
```

In most cases, a base DN was not set up on one of the directory servers or the backend is disabled.

13. Next, the prompt asks if you want to set up entry balancing using the Directory Proxy Server. Press Enter to accept the default (no), since we are not setting up replication in an entry-balanced environment in this scenario. For more information, see the *PingDirectoryProxy Server Administration Guide*.

```
Do you plan to configure entry balancing using the Directory Proxy Server?
 (yes / no) [no]:
```

14. Next, enter the replication port for the first replica (default, 8989). The port must be free.
15. If the first server did not have a pre-defined location setting, `dsreplication` will prompt you to enter a location. Press Enter to accept the default (yes) to set up a Location for the first server. Enter the name of the server's location.

```
The first server has not been configured with a location.
Assigning a location to each server in the replication topology reduces
network traffic in multi-site deployments. Would you like to set the
location in the first server? (yes / no) [yes]

The location of the first server: Austin
```

16. Repeat the previous steps for the second directory server. Again, if you did not pre-define a location setting for the second server, you will be prompted to enter this information.
17. At this time, set up the Global Administrator user ID (default is "admin") and a password for this account. The Global Administrator user ID manages the directory servers used in the replication topology.

```
Specify the user ID of the global administrator account
that will be used to manage the Ping Identity Directory Server
instances to be replicated [admin]:

Password for the global administrator:
Confirm Password:
```

18. Return to the Replication Main Menu and enter the number corresponding to initializing data over the network.
19. On the Initialize Replica Data over the Network menu, select Initialize to initialize data on a single server, and then enter `c` to continue.
20. Next, specify a server in the replication topology. For this example, enter the host name or IP address, LDAP connection type, LDAP port, Global Admin user ID and password of the first server.
21. Next, select the source server that is hosting the data to which the target server will be initialized. For this example, select the first server, since the sample dataset has been loaded onto this server.

22. Next, select the base DN that will be initialized. In most cases, the base DN for the root suffix will be replicated. In this example, `dc=example,dc=com`.
23. Next, select the second server in this example that will have its data initialized, and then enter the Global Admin user ID and password for the target server. Any data present on the target server will be over-written.
24. Press Enter to confirm that you want to initialize data on the target server. When completed, you should see "Base DN initialized successfully."

```
Initializing the contents of a base DN removes all the existing contents of
that base DN. Do you want to remove the contents of the selected base DNs on
 server
server2.example.com:1389 and replace them with the contents of server
server1.example.com:1389? (yes / no) [yes]:
```

25. On the Initialize Replica Data Over Network menu, enter `b` to back out one level to the main menu. Then, on the Replication Main menu, enter the number to view the replication status.

```
     --- Replication Servers: dc=example,dc=com ---
Server                   : Location : Conflict Entries : Backlog : Recent
 Change Rate
-------------------------:----------:------------------:---------:--------------------
ds1 (example.com:1389)   : austin   : 0                : 0       : 0
ds2 (example.com:1389)   : austin   : 0                : 0       : 0
```

A Deployment with Non-Interactive dsreplication

This example will create a four-server topology spanning two data centers. The four servers are already installed and have locations Austin and Budepest defined.

Note: When enabling or initializing servers, the `--topologyFilePath` option can be used with `dsreplication` to specify a file with a series of hosts and ports available in the topology that can be used as source servers. This option is used in place of specifying host 1, port1. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available for replication. This file is generated with the `manage-topology export` command.

To Deploy with Non-Interactive dsreplication

1. Generate the sample data using the `make-ldif` tool on the first server.

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-10K.template \
  --ldifFile ldif/10K.ldif
```

2. Select a server from which to import data, and to be the source for future initialization to other servers. Stop this server, import the sample LDIF and start again, or perform a task-based `import-ldif` with the connection options.

```
$ bin/stop-server
$ bin/import-ldif --backendID userRoot --ldifFile ldif/10K.ldif
$ bin/start-server
```

3. Enable replication by choosing a specific server and running `dsreplication enable` three times. For the first invocation, create the replication topology administrator with the name `admin`.

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 austin02.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt
```

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 budapest01.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt

$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 budapest02.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt
```

4. Initialize the other servers (the dc=example,dc=com replicas) from the server that had the data imported with `import-ldif`. To minimize the WAN transfers, initialize `budapest02` from `budapest01`.

```
$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
  --hostDestination austin02.example.com --portDestination 1389 \
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt

$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
  --hostDestination budapest01.example.com --portDestination 1389 \
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt

$ bin/dsreplication initialize --hostSource budapest01.example.com --
portSource 1389 \
  --hostDestination budapest02.example.com --portDestination 1389 \
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt
```

5. View the state of replication.

```
$ bin/dsreplication status --adminPassword password --no-prompt --
displayServerTable --showAll
```

To Use dsreplication with SASL GSSAPI (Kerberos)

This example procedure assumes that you have configured SASL GSSAPI on all servers in the replication topology and that they are working properly.

The Directory Server's utilities all support SASL GSSAPI options for systems using Kerberos as its main authentication mechanism. The following procedure shows how to use `dsreplication` with SASL GSSAP to set up a new `replication.admin` identity, while enabling replication on a server. The following are important points about the configuration:

- A separate Kerberos identity is required to manage replication. Existing Kerberos credentials can be used to interact with the server when enabling replication and creating the new identity.
- The new identity, such as replication.admin, must not exist as the cn or uid value under any public base DN.

1. Set the LDAP Connection Handler to explicitly listen on the server's hostname address.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --remove listen-address:0.0.0.0 --add listen-address:host.example.com
```

2. Update the identity mapper to have `cn=topology,cn=config` included in the list of base DNs and to add the `cn` attribute to match attributes. This step is required to map the admin account to the Kerberos realm.

```
$ bin/dsconfig set-identity-mapper-prop \
  --mapper-name "Regular Expression" \
  --add match-attribute:cn \
  --set "match-base-dn:cn=topology,cn=config" \
  --set match-base-dn:dc=example,dc=com
```

3. Invoke replication enable, authenticating as the existing kerberos `authid`. Note that no bind DNs and passwords are required to authenticate because we are using SASL binding. However, the new replication admin user requires a password at creation time, so we recommend using a strong random password. Once SASL is working, you will no longer have to provide this random password. Also note that we are forcing the use of the ticket cache, so make sure you have properly authenticated as `ds.admin` from your local host and the ticket is not expired in the cache.

```
$ kinit -p ds.admin
$ bin/dsreplication enable \
  --host1 server1.example.com --port1 1389 --replicationPort1 1989 \
  --host2 server2.example.com --port2 2389 --replicationPort2 2989 \
  --baseDN dc=example,dc=com \
  --adminUID replication.admin --adminPassword strongPassword \
  --saslOption1 mech=gssapi --saslOption1 authid=ds.admin@EXAMPLE.COM \
  --saslOption1 useTicketCache=true --saslOption1 requireCache=true \
  --saslOption2 mech=gssapi --saslOption2 authid=ds.admin@EXAMPLE.COM \
  --saslOption2 useTicketCache=true --saslOption2 requireCache=true
```

4. Use `dsreplication initialize` to initialize data on remote server.

```
$ kinit -p replication.admin
$ bin/dsreplication initialize \
  --hostSource server1.example.com --portSource 1389 \
  --hostDestination server2.example.com --portDestination 2389 \
  --baseDN dc=example,dc=com \
  --saslOption mech=GSSAPI \
  --saslOption authID=replication.admin@EXAMPLE.COM  \
  --no-prompt
```

5. The temporary `userPassword` can now be deleted from the `replication.admin` entry. Create a file called `remove-password.ldif` with these contents:

```
dn: cn=replication.admin,cn=Administrators,cn=topology,cn=config
changetype: modify
delete: userPassword

```

6. Apply the modifications using `ldapmodify`:

```
$ ./ldapmodify --filename remove-password.ldif -o mech=GSSAPI
-o authid=replication.admin@example.com \
  --saslOption useTicketCache=true \
  --hostname host.example.com --port 1389 \
  --noPropertiesFile
```

7. Check the topology's status by running `dsreplication status`. The `--saslOption useTicketCache=true` and `--saslOption requireCache=true` properties, instead of providing a password, for all `dsreplication` commands after properly creating the admin accounts and mappers.

```
$ bin/dsreplication status \
  --saslOption mech=gssapi \
  --saslOption authid=replication.admin@EXAMPLE.COM \
  --saslOption useTicketCache=true --saslOption requireCache=true \
  --hostname host.example.com --port 1389 \
  --no-prompt
```

Configuring Assured Replication

The PingDirectory Server's replication mechanism is based on the eventual-consistency model, which is a loosely-connected topology that propagates updates to other servers without waiting for their corresponding replication response messages. As a result, there is a small window of time where updates are not all present on the replicas as the changes are replicating through the system. There are, however, deployments that require *immediate* access to replicated data. In such cases, administrators can configure *Assured Replication*, which ensures that replication has completed *before* the update response is returned to the client.

From the LDAP client's perspective, assured replication has no bearing on the result code of the operation, just on the time in which it takes to receive the response for those requests in which replication assurance is applied. Detailed information regarding assurance processing is available to an LDAP client with awareness of the assured replication control.

The assured replication mechanism takes advantage of server location to distinguish between local and remote servers to allow different policies to apply. For example, a common assurance approach is to respond to a client update after all servers in the same location have applied the update, and one or more servers in remote locations have received the update. In addition, the level of assurance applied to each operation can be explicitly requested by the client and/or specified by the server configuration using the Replication Assurance Policy.

Assured replication is supported by client requests directly to PingDirectory Server and/or through a PingDirectoryProxy Server.

About the Replication Assurance Policy

Assured replication uses a *Replication Assurance Policy* to define the properties needed to ensure that replication has satisfactorily completed before the update response is returned to the client. Multiple policies can be defined but only one policy is matched with a client update request. Each policy contains an evaluation order index which, together with an optional request and connection criteria, provides flexibility in matching a policy to request.

The Replication Assurance Policy defines local and remote assurance *levels*. A *level* defines how rigorous the policy should be when waiting for propagation of updates, while *local* and *remote* distinguish servers in the same location versus servers in remote locations. Although optional, it is recommended that request or connection criteria be associated with a policy to apply replication assurance appropriately.

The Directory Server contains a Default Replication Assurance Policy that is enabled but has no assurance levels assigned. In addition to using the Default Replication Assurance Policy, any number of policies can be created and enabled. When a client request is received, the server iterates through the list of enabled policies according to each policy's *evaluation-order-index* property. A smaller *evaluation-order-index* value (e.g., 1) has precedence over policies with larger *evaluation-order-index* values (e.g., 2, 3, 4, etc.). The *evaluation-order-index* values do not need to be contiguous. The first policy that matches a request is associated with the operation.

The Replication Assurance Policy, which is defined on the PingDirectory Server and not on the PingDirectoryProxy Server, has the following properties:

- **evaluation-order-index**. Determines the evaluation order (the smaller value has precedence) among multiple Replication Assurance Policies that match against an operation.
- **local-level**. Specifies the assurance level used to replicate to local servers. A local server is defined as a server with the same `location` property value as set in the global configuration. The `local-level` property must be set to an assurance level as strict as the `remote-level` property. For example, if the `remote-level` is set to "`processed-all-remote-locations`," then the `local-level` property must be "`processed-all-servers`."

  None. Replication to any local server is not assured.

  received-any-server. At least one available local server must receive a replication update before a response is sent to the client.

  processed-all-servers. All available local servers must complete replay of the replication update before the response is sent to the client. If a singular server is enabled, or only one server is available for a particular location, `processed-all-servers` will return a value of `false`.

- **remote-level**. Specifies the assurance level used to replicate to remote servers. A remote server is defined as a server that has a different `location` property value as set in the global configuration.

  None. Replication to any remote server is not assured.

  received-any-remote-location. At least one server at any available remote location must receive a replication update before a response is sent to the client.

  received-all-remote-locations. All available remote servers must receive a replication update before the response is sent to the client.

  processed-all-remote-servers. All available servers at all locations must complete replay of the replication update before the response is sent to the client. If a single server is enabled, or only one server is available for a particular location, `processed-all-remote-servers` will return a value of `false`.
- **timeout**. Specifies the maximum length of time to wait for the replication assurance requirements to be met before timeout and replying to the client.
- **connection-criteria**. Specifies connection criteria used to indicate which operations from clients matching this criteria use this policy. If both connection criteria and request criteria are specified for a policy, then both must match an operation for the policy to be assigned.
- **request-criteria**. Specifies request criteria used to indicate which operations from clients matching this criteria use this policy. If both connection criteria and a request criteria are specified for a policy, then both must match an operation for the policy to be assigned.

Servers in a replication topology are not required to share a homogeneous set of policies; you can configure the Replication Assurance Policies differently on the replicas in a topology. For example, if you configure server A to match add operations to a `processed-all-servers` assurance level, and server B to match add operations to a local `received-any-server` level, then add operations received on server A will have the assurance level of `processed-all-servers` and add operations received on server B will have the assurance level of `received-any-server`.

For more detailed information, see the *PingDirectory Server Configuration Reference Guide*.

Points about Assured Replication

The following are some points when implementing Assured Replication:

- **Client Controls**. The client may optionally include an assured replication request control with each operation. This control allows the client to specify bounds on assurance levels and/or override the timeout assigned by the matching replication assurance policy for the associated operation. The server always honors these request controls. See *About the Assured Replication Controls* for more information.
- **Directory Proxy Server**. Replication assurance policies are not supported on the PingDirectoryProxy Server. Replication client controls are passed through to the underlying Directory Server backend.
- **Schema backend Replication**. The schema backend is not supported by Assured Replication. Replication assurance policies that include criteria to match this backend will be rejected.
- **Backward Compatibility**. Server versions that support assured replication are backwards-compatible with prior versions that do not support assured replication.
- **WAN-Friendly Replication**. Assured replication functions independently from WAN-Friendly Replication and the notion of WAN Gateways.
- **Global Configuration Properties**. The Directory Server provides two configurable global configuration properties that determine the timing of the assurance source and maximum number of replication backup updates to be recognized as an available source.

  - **replication-assurance-source-timeout-suspend-duration**. Specifies the amount of time a replication assurance source will be suspended from assurance requirements if it experiences an assurance timeout. While suspended, the source will be excluded from assurance requirements for all operations originating on this Directory Server. This avoids the situation of repeated timeouts caused by degraded or offline servers. Default is 10 seconds.
  - **replication-assurance-source-backlog-fast-start-threshold**. Specifies the maximum number of replication backlog updates a replication assurance source can have and be immediately recognized as an available source. If a source connects to this server with more than the configured threshold backlog updates, it will be excluded

from assurance requirements for all operations originating from this Directory Server until it completes at least one assurance successfully (i.e. this Directory Server receives an update acknowledgement message from it within the timeout window). Default is 1000.

To Configure Assured Replication

This example illustrates configuring a variety of assured replication policies. In practice it's common for all servers to have the same policy. The following example assumes that three servers are configured on localhost, on ports 1389, 2389 and 3389. Note that each server has a default Replication Assurance Policy with no assurance levels set.

1. On server 1, use `dsconfig` to create request criteria for add operations. This request criteria will be used to match any add operation with the Replication Assurance Policy that will be configured in the next step.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Adds \
  --type simple \
  --set operation-type:add
```

2. On server 1, set up the Replication Assurance Policy to make all add operations assured with a level of `processed-all-servers`, which indicates that all local servers in the topology must complete replay of the replication update before the response is sent to the client. Specify the Adds request criteria configured in the previous step.

```
$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Adds Processed All Locally" \
  --set evaluation-order-index:1 \
  --set local-level:processed-all-servers \
  --set "timeout:500ms" \
  --set request-criteria:Adds
```

3. On server 1, repeat the previous two steps for modify operations. The Replication Assurance Policy "Mods Received Any Locally" ensures that at least one available local server must receive a replication modify update before a response is sent to the client.

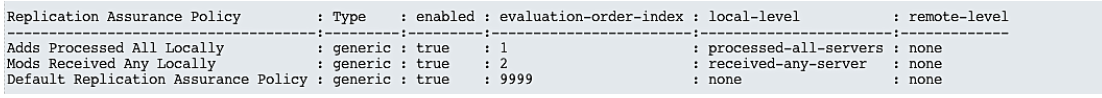
```
$ bin/dsconfig create-request-criteria \
  --criteria-name Mods \
  --type simple \
  --set operation-type:modify

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Mods Received Any Locally" \
  --set evaluation-order-index:2 \
  --set local-level:received-any-server \
  --set "timeout:500ms" \
  --set request-criteria:Mods
```

4. On server 2, repeat steps 1-3 to set up the Adds and Mods request criteria and its respective Replication Assurance Policy.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Adds \
  --type simple \
  --set operation-type:add

$ bin/dsconfig create-request-criteria \
  --criteria-name Mods \
  --type simple \
  --set operation-type:modify

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Adds Received Any Locally" \
  --set evaluation-order-index:1 \
  --set local-level:received-any-server \
  --set "timeout:500ms" \
  --set request-criteria:Adds

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Mods Processed All Locally" \
  --set evaluation-order-index:2 \
  --set local-level:processed-all-servers \
  --set "timeout:500ms" \
  --set request-criteria:Mods
```

5. Leave server 3 with the default Replication Assurance Policy configured with no assurance levels or criteria. In practice it is common for all servers to have the same assurance levels or criteria.
6. On server 1, list the policies on the server using the `dsconfig` command to confirm that they exist on the server.

```
$ bin/dsconfig list-replication-assurance-policies
```



```
Replication Assurance Policy         : Type    : enabled : evaluation-order-index : local-level           : remote-level
-------------------------------------:---------:---------:------------------------:-----------------------:-------------
Adds Processed All Locally           : generic : true    : 1                      : processed-all-servers : none
Mods Received Any Locally            : generic : true    : 2                      : received-any-server   : none
Default Replication Assurance Policy : generic : true    : 9999                   : none                  : none
```

7. Repeat the previous step on server 2 and server 3. Server 3 should only show the Default Replication Assurance Policy.
8. Check the Replication Assurance counters on all servers before any add or modify operation using `ldapsearch`. They should be set to zero. These counters are on the replica server, which is where the policy is matched and assigned. On server 1, run the following command:

```
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-local-completed-normally: 0
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

9. Check the Replication Summary table on all of the servers. For example, on server 1, run the following command:

```
$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 0
replication-assurance-local-completed-normally: 0
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

10. Add an entry to the server 1 Directory Server. Check that the counters matched the newly added entry to the "Adds Processed All Locally" Policy and that it completed assured.

```
$ bin/ldapmodify --filename add-user.ldif --defaultAdd
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 1
replication-assurance-local-completed-normally: 1
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
replication-assurance-policy-matches: Adds Processed All Locally: 1
replication-assurance-policy-matches: Default Replication Assurance Policy:
 0
replication-assurance-policy-matches: Mods Received Any Locally: 0
replication-assurance-local-level-uses: processed-all-servers: 1
replication-assurance-remote-level-uses: none: 1

$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 1
replication-assurance-local-completed-normally: 1
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

11. Perform a modify of an entry under dc=example,dc=com on server 1. Check that the counters matched the modify operation to the "Mods Processed All Locally" policy and that the operations completed assured.

```
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 2
replication-assurance-local-completed-normally: 2
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
replication-assurance-policy-matches: Adds Processed All Locally: 1
replication-assurance-policy-matches: Default Replication Assurance Policy:
 0
replication-assurance-policy-matches: Mods Received Any Locally: 1
replication-assurance-local-level-uses: processed-all-servers: 1
replication-assurance-local-level-uses: received-any-server: 1
replication-assurance-remote-level-uses: none: 2

$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 2
replication-assurance-local-completed-normally: 2
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

You have successfully configured Assured Replication.

About the Assured Replication Controls

The LDAP SDK for Java provides an implementation of an LDAP control that can be included in add, bind, modify, modify DN, and certain extended requests to indicate the level of replication assurance desired for the associated operation. The OID for this control is 1.3.6.1.4.1.30221.2.5.28, and may have a criticality of either TRUE or FALSE.

For specific details, see the LDAP SDK javadoc for the `AssuredReplicationRequestControl` class.

Managing the Topology

The following sections describe common topology management operations.

Note: When enabling or disabling replication within a topology that contains multiple product versions, the `dsreplication` tool must be run from the server root location of a member of the topology that has the oldest product version.

To Add a Server to the Topology

The following steps assume an existing directory server topology. The commands are identical for initial enable between two servers, where one server contains data for the replication domain stored in the `userRoot` backend. If database encryption is being used on the servers in the topology, it is important that the server being initialized has a copy of the `encryption-settings` backend from the source server.

1. A majority of servers (more than 50%) in the topology and the new server, should be online.
2. Enable replication for the base DN, or base DNs, using an existing server as `host1` and the new server as `host2`.

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 austin03.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt
```

3. Optionally, compare the configurations between the two hosts used in the `dsreplication enable` command. Make sure settings are consistent across the topology and are also consistent with the new system:

```
$ bin/config-diff --sourceLocal --targetHost austin03.example.com
--targetBindDN "cn=directory manager" --targetBindPassword pass --targetPort
 1389
```

Disabling Replication and Removing a Server from the Topology

When removing a server from the topology, the remaining servers need to be made aware of the change. If the server to be removed is online, then one invocation of `dsreplication disable` is all that is necessary. If the server to be removed is offline, then two steps are required: `remove-defunct-server` from another server in the topology, and `remove-defunct-server` on the offline server to be removed. Similar to the `enable` command, more that 50% of servers not being removed from the topology need to be online during the process.

If there are additional servers that are offline and can not be online while the offline server is being removed, then it's important to make a distinction between offline servers that are permanently offline, and those that are temporarily offline. If servers are permanently offline, they should also be removed with `remove-defunct-server`. If servers are temporarily offline, once they are online, they will automatically update. The following examples show the steps in more detail:

- **Removing a server that is still online**. The `dsreplication disable` command can be run from any server, but a majority of servers in the topology need to be online.

```
$ bin/dsreplication disable --hostname austin03.example.com --port 1389 \
              --baseDN dc=example,dc=com --adminUID admin --adminPassword
 password \
              --no-prompt
```

- **Removing a server that is offline**.The `remove-defunct-server` tool can be run against any server not being removed from the topology. More than 50% of servers in the topology should be online. The `remove-defunct-server` tool can be issued after setting the JVM property `"com.unboundid.connectionutils.LdapResponseTimeoutMillis"` to change the default ten minute time out for each server to be taken out of rotation. If there are multiple servers to be removed, this can speed up the process.

```
$ bin/remove-defunct-server --serverInstanceName austin01 \
  --bindDN "cn=Directory Manager" --bindPassword password
```

  Run the `remove-defunct-server` tool on each server removed from the topology to remove any topology references.

```
$ bin/remove-defunct-server --no-prompt
```

Replacing the Data for a Replicating Domain

In the rare event that the data for the entire replication domain (such as the backend) needs to be replaced, perform the following steps:

To Replace the Data

1. With all servers online, the `dsreplication pre-external-initialization` command must be run once against any server in the topology. This stops replication for the domain. No writes are made by clients to any of the servers.

```
$ bin/dsreplication pre-external-initialization --hostname
 austin01.example.com \
   --port 1389 --baseDN dc=example,dc=com --adminUID admin \
   --adminPassword password --no-prompt
```

2. Using `import-ldif`, replace the data for `dc=example,dc=com`. Make sure that the input LDIF is free of any replication attributes by using the `--excludeReplication` option. The `--overwriteExistingEntries` option is necessary to overwrite the existing data for the domain. For example, to perform the `import-ldif` with the server offline:

```
$ bin/import-ldif --ldifFile new-data.ldif --backendID userRoot --
excludeReplication --overwriteExistingEntries
```

3. Initialize the other servers in the topology with `dsreplication initialize`, using the server which has the new data as the source host. For example:

```
$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
   --hostDestination budapest01.example.com --portDestination 1389 \
   --adminUID admin --adminPassword password --baseDN dc=example,dc=com \
   --no-prompt
```

4. Run `dsreplication post-external-initialization` once from any server in the topology. All servers in the topology must be online:

```
$ bin/dsreplication post-external-initialization --hostname
 austin01.example.com \
   --port 1389 --baseDN dc=example,dc=com --adminUID admin \
   --adminPassword password --no-prompt
```

Advanced Configuration

The following sections are advanced configuration procedures that may be appropriate for your company's deployment.

Changing the replicationChanges DB Location

You can change the `replicationChanges` DB location if on-disk space issues arise. The replication changelog database can live outside `<server-root>` and be placed in another location on the filesystem. In that case, you must specify the absolute path of the replication changelog directory.

To Change the replicationChanges DB Location

1. Use `dsconfig` to change the database location for the replication changelog, which by default is at `<server-root>/changelogDb`. The following command sets the replication changelog backend to `<server-root>/data/directory/changelogDB` . Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set "replication-db-directory:/data/directory/changelogDb" \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

2. Stop the server, move the DB files, and then restart the server.

```
$ bin/stop-server
$ mv changelogDb /data/directory
$ bin/start-server
```

Modifying the Replication Purge Delay

The replication purge delay specifies the period after which the directory server purges changes on the replication server database. Any change that is older than the purge delay is removed from the replication server database regardless of whether the change has been applied.

Currently, the PingDirectory Server sets the default purge delay to one day. Administrators can change the default purge delay using the `dsconfig` tool. To ensure proper replication processing, you must have the same purge delay value set for all replication servers in the topology.

To Modify the Replication Purge Delay

- Use `dsconfig` to change the purge delay. The property accepts time units of seconds (s), minutes (m), hours (h), days (d), or weeks (w). The following command is entered on the command line and changes the purge delay from the default one day to two days.

  In `dsconfig` interactive mode, open the **Advanced objects** menu. Select **Replication Server**. Select the replication synchronization provider, and then select the option to change the replication purge delay.

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set "replication-purge-delay:2 d"
```

Configuring a Single Listener-Address for the Replication Server

By default, the replication server binds the listening ports to all available interfaces of the machine. To bind the listener to a specific address, the address must be the hostname provided when replication is enabled and the `listen-on-all-addresses` property must be set to `FALSE`.

The replication server's configuration entry already stores a host name for itself so that it can resolve the address and specify it during the socket bind. If the server information is missing from the system, an error message will be generated with instructions on specific address binding. You can use the `dsconfig` tool to change the value of the `listen-on-all-addresses` property from TRUE (default) to FALSE.

To Configure a Replication Server to Listen on a Single Address

1. Create a new directory serverinstance with replication enabled on port 8989.
2. Run `netstat` to see the ports bound for listening on port 8989. Notice that *.8989 means that it is listening on all addresses.

```
$ netstat -an | grep LISTEN | grep 8989
```

3. Run `dsconfig` to disable listening on all addresses for the replication server.

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set listen-on-all-addresses:false
```

4. Run `netstat` again to see the ports bound for listening on port 8989. Notice that `<address>.8989` (for example, 10.8.1.211.8989) means that it is listening on the one address.

Monitoring Replication

Replication in the PingDirectory Server can be monitored the following ways:

- The `dsreplication` status subcommand displays basic information about the replicated base DNs, the number of entries replicated as well as the approximate size of replication backlogs at each Directory Server.
- The more detailed information about the state of replication can be obtained via the information exposed in its monitoring framework under `cn=monitor`. Administrators can monitor their replication topologies using several tools and protocols: the Data Metrics Server, SNMP, LDAP, JMX, or through the Administrative Console. See Managing Logging and Monitoring.
- The Periodic Stats Logger plug-in allows collecting replication statistics for profiling server performance. For more information, see Profiling Server Performance Using the Periodic Stats Logger.

Monitoring Replication Using cn=monitor

The `cn=monitor` branch has a number of entries that store the replication state of a topology.

- **Direct LDAP Server <baseDN> <host name:port> <serverID>**. Defines an LDAP server that is directly connected to the replication server that you are querying. The information in this entry applies to the replication server local to the `cn=monitor` entry. For detailed information, see Summary of the Direct LDAP Monitor Information.
- **Indirect LDAP Server <baseDN> <serverID>**. Defines an LDAP server that is connected to another replication server in the topology. While this server is connected to the same topology, it is not connected to the replication server being queried. For detailed information, see Summary of the Indirect LDAP Server Monitor Information.

- **Remote Repl Server <baseDN> <host name:port> <serverID>**. Defines a remote replication server that is connected to the local replication server. Information in this entry is in respect to the Replication Server local to the `cn=monitor` branch. For detailed information, see Summary of the Remote Replication Server Monitor Information.
- **Replica <baseDN>**. Stores information for an instance of the replicated naming context— also known as the replica—with respect to the Directory Server and its communication with a replication server. The Replica information is what is responsible for sending and receiving changes from the replication servers. For detailed information, see Summary of the Replica Monitor Information.
- **Replication Server <replPort> <serverID>**. Shows the information specific to the Replication Server running, for example, on the replication port <replPort> with a server ID of <serverID>. This entry defines the replication server. For detailed information, see Summary of the Replication Server Monitor Information.
- **Replication Server Database <baseDN> <serverID>**. Shows information for the changelog table of replica (suffix) in the replication server. As the Replication Server receives updates from the Directory Server, it records those changes in the changelog table. For detailed information, see Summary of the Replication Server Database Monitor Information.
- **Replication Server Database Environment <baseDN>**. Shows the information for the database environment for the replication server backend plus the total number of records added to and deleted from the database. For detailed information, see Summary of the Replication Server Database Environment Monitor Information.
- **Replication Summary <baseDN>**. Shows summary information on the replication topology and the state for a particular base DN. For detailed information, see Summary of the Replication Summary Monitor Information.
- **Replication Changes Backend**. Shows the backend information for all replication changes. For detailed information, see Summary of the Replication Changes Backend Monitor Information.
- **Replication Protocol Buffer**. Shows the state of the buffer (initially 4k) for protocol operations, which is stored in thread local storage. For detailed information, see Summary of the Replication Protocol Buffer Monitor Information.

Replication Best Practices

The following are recommended best practices related to replication based on our observations in actual production environments.

About the dsreplication Command-Line Utility

The following points involve some security practices as applies to replication. For specific questions, please contact your authorized support provider.

- **Developing Scripts**. The `dsreplication` utility maintains the history of executed `dsreplication` commands with the full command-line arguments in the `logs/tools/dsreplication.history` file. The recorded commands may be used to develop scripts to set up and configure replication.

  Scripts invoking the `dsreplication` utility in non-interactive mode should check the return code from the dsreplication process. A non-zero return code indicates some sort of failure.

  If output messages from the `dsreplication` utility are not desired, use the --quiet option to suppress them.

  The utility, by default, fails if one or more warnings are issued during the command execution. Warnings can be suppressed using the --ignoreWarnings option. For example, this option is required when using `dsreplication` with non-fully-qualified hostnames (for example, localhost), otherwise `dsreplication` will fail. In production environments, use of this flag is strongly discouraged.

  The `dsreplication` utility also provides an --ignoreLock option that specifies that the tool should continue processing in non-interactive mode or in scripts even if the replication topology has been locked by another invocation of `dsreplication`. However, this option should be used with caution.
- **Concurrent Use**. With the exception of the `dsreplication` status subcommand, the `dsreplication` subcommands cannot be executed concurrently. The command-line utility locks the replication topology at one or more servers to prevent accidental configuration changes caused by multiple `dsreplication` subcommands running at the same time. It is best to avoid concurrent configuration changes in general.

- **Status**. The `dsreplication` status subcommand requires the Replication Servers to provide monitoring information. This can lead to a delay before the output of `dsreplication` status is displayed. By default, `dsreplication` will display the status for all replicated domains (with the exception of the special domains of the schema and the server registry).

  It is recommended to select a particular base DN for `dsreplication` status if multiple base DNs are configured for replication.

  It is also recommended to avoid invoking `dsreplication` status too often (more than once every 15 seconds) or from multiple locations at the same time. Some of the information displayed by `dsreplication` status is based on monitor information that is not refreshed every time the monitor is queried.

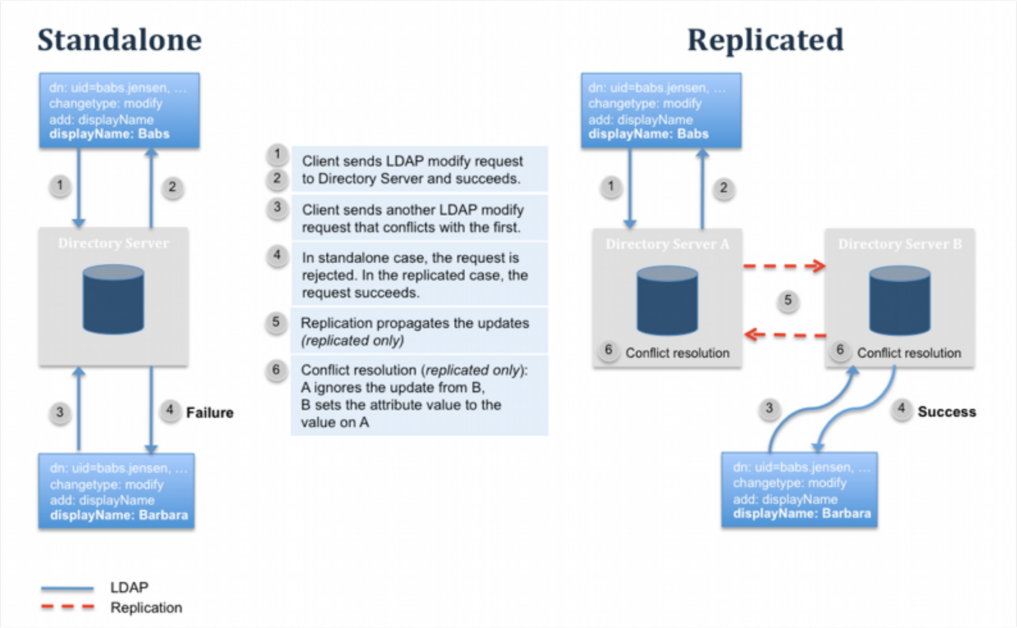
  The status subcommand should not be used for collecting performance metrics. It is best to rely on replication-related information captured by the Periodic Stats Logger Plug-in.
- **Topology Tasks**. The `dsreplication` tool allows specifying more than one server in a topology to act as the host for other servers for the enable and initialize actions. The `--topologyFilePath` option can be used to specify a file with a series of hosts and ports in the topology. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available. This file is generated with the `manage-topology export` command.

Replication Conflicts

This section provides more in-depth information on replication conflicts than presented in earlier sections, so that administrators can understand the mechanisms and possible scenarios behind these conflicts.

Updates to Directory Server entries in a replication topology may happen independently, since replication guarantees only eventual consistency, not strong consistency. The eventual consistency model also means that conflicting changes can be applied at different directory server instances. In most cases, the Directory Server is able to resolve these conflicts automatically and in a consistent manner (i.e., all directory server instances in a replication topology will resolve each and every conflict the same way). However, in some scenarios, as seen below, manual administrative action is required. For any of these unresolved conflicts, the administrator is notified via administrative alerts.

On a high-level, the conflict resolution algorithm tries to resolve conflicts as if the operations causing the conflict in a distributed environment has been applied to a single directory server instance. For example, if the same entry is added to two different directory server instances at about the same time, then once these operations have been replicated, both directory servers will keep only the entry that was added first. The following figure highlights the differences between standalone versus replicated environments.



1 2 Client sends LDAP modify request to Directory Server and succeeds.

3 Client sends another LDAP modify request that conflicts with the first.

4 In standalone case, the request is rejected. In the replicated case, the request succeeds.

5 Replication propagates the updates *(replicated only)*

6 Conflict resolution (*replicated only*): A ignores the update from B, B sets the attribute value to the value on A

Figure 12: Conflicting Operations in Standalone versus Replicated Environments

Types of Replication Conflicts

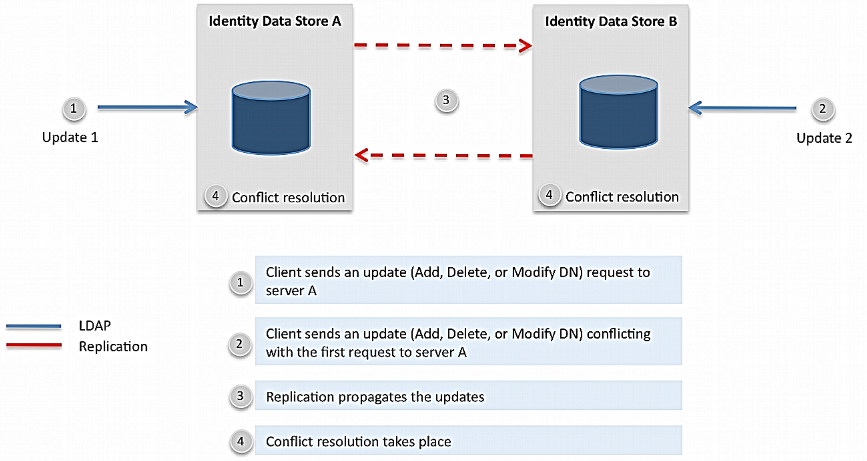
There are fundamentally two types of replication conflicts: naming and modification conflicts. Naming conflicts include operations that cause conflicts with the naming (DN) of the existing or new entries, while modification conflicts include operations that result in conflicts in the modification of attributes.

Naming Conflict Scenarios

For all of the naming conflict scenarios in the table below, assume the folloing:

- Update 1 was applied at Directory Server 1
- Update 2 was applied at Directory Server 2
- Update 1 occurred shortly before Update 2, so that Directory Server 2 received Update 1 after Update 2 was applied

The naming conflict scenario is illustrated in the following figure:



1 Client sends an update (Add, Delete, or Modify DN) request to server A

2 Client sends an update (Add, Delete, or Modify DN) conflicting with the first request to server A

3 Replication propagates the updates

4 Conflict resolution takes place

Figure 13: Naming Conflict Scenario

The following table shows the result of a modification conflict depending on the type of updates that occurs. The code does not compare change sequence numbers (CSNs) but applies operations in the order they were received. This may lead to inconsistent replays.

Table 46: Naming Conflict Scenarios

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at Directory Server 2 When Update 1 is received
Modify	Delete	Yes	Modify is discarded.
Modify	Modify DN	Yes	New entry is located based on the entryUUID and Modify is applied to the renamed entry.
Delete	Delete	Yes	Delete operation is ignored.
Delete	Modify DN	Yes	Delete operation is applied to the renamed entry.
Delete of A	Add of B under A	Yes	Entry B is renamed and Entry A is deleted.
Modify DN	Delete of the same entry targeted by the Modify DN	Yes	Modify DN operation is ignored.
Modify DN with a new parent	Delete of parent	No	The entry targeted in the Modify DN operation is marked as a conflict entry.
Modify DN with a new parent	Modify DN of the parent	Yes	The entry will be moved under the new DN of the parent.
Modify DN of A with new DN B	Modify DN of C with new DN B	No	A and B will be conflict entries.
Add A	Modify DN of the parent of A	Yes	The entry is added under the new DN of the parent.
Add A	Delete of the parent of A	No	The added entry is marked as a conflict entry.
Add A	Add A with same set of attributes	Yes	The entryUUID of the incoming Add operation applied to the existing entry.
Add A	Add A with different set of attributes (or values)	No	The existing entry is marked as a conflicting entry and the incoming Add is executed.

Modification Conflict Scenarios

Modification conflicts are always resolved automatically and no manual action is required. The LDAP Modify operation allows the following modification types:

Add of one or more values
Delete of one or more values or the entire attribute
Replacement of all values

Replication does not currently support the increment LDAP modification type.

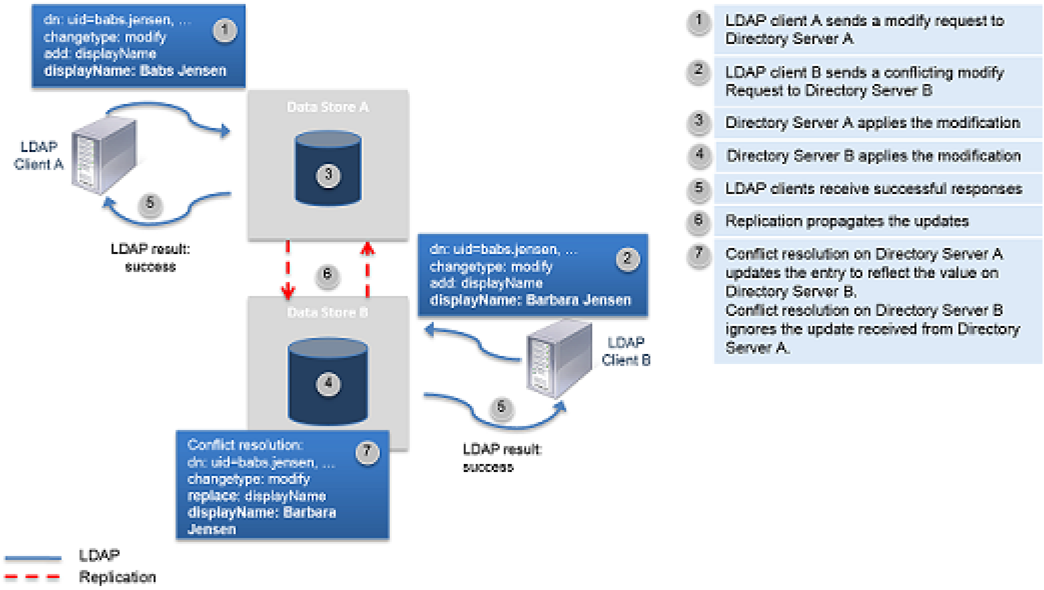
For all of the operations in the table below, assume the following:

LDAP Modify 1 was applied at Directory Server 1

LDAP Modify 2 was applied at Directory Server 2

LDAP Modify 1 occurred shortly before LDAP Modify 2, so that Directory Server 2 received LDAP Modify 1 after LDAP Modify 2 was applied.

The modification conflict scenario is illustrated in the figure below:



1 LDAP client A sends a modify request to Directory Server A

2 LDAP client B sends a conflicting modify Request to Directory Server B

3 Directory Server A applies the modification

4 Directory Server B applies the modification

5 LDAP clients receive successful responses

6 Replication propagates the updates

7 Conflict resolution on Directory Server A updates the entry to reflect the value on Directory Server B. Conflict resolution on Directory Server B ignores the update received from Directory Server A.

Figure 14: Modification Conflict Scenario

The following table shows the result of a modification conflict depending on the type of updates that occurs:

Table 47: Modification Conflict Scenarios

Modify 1	Modify 2	Result of Conflict Resolution at Directory Server 2 When Modify 1 is received
Add of a single-value attribute	Add of the same attribute with a different value	Incoming Modify is ignored.
Delete of a single-valued attribute	Replacement of the value of the same attribute	Incoming Delete is ignored.
Replacement of a single-valued attribute	Delete of the same attribute	Incoming Replacement is ignored.
Delete some values from a multi-valued attribute	Delete some values from a multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Delete of the same multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Add the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Delete value X from the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Add value Y to the same multi-valued attribute	Delete of value X is applied.
Delete value X from a multi-valued attribute	Delete value Y from the same multi-valued attribute	Delete of value X is applied.

Modify 1	Modify 2	Result of Conflict Resolution at Directory Server 2 When Modify 1 is received
Delete value X from a multi-valued attribute	Replace all values of the same multi-valued attribute	Incoming Delete is ignored.
Add of values X and Y to a multi-valued attribute	Delete value X from the same multi-valued attribute	Only value Y is added.
Delete value X from a multi-valued attribute	Add of values X and Y to the same multi-valued attribute	Incoming Delete is ignored.

Troubleshooting Replication

The following sections provide information to troubleshoot your replication deployment.

Recovering a Replica with Missed Changes

If a server has been offline for a period of time longer than the replication purge delay, the `dsreplication initialize` command must be performed to bring the replica into sync with the topology.

Server startup is the only time missed changes are detected. A missed change is a change that the replica detects that it needs, but which cannot be found within any other replication server's replicationChanges backend (stored in the path server root `/changelogDb`). If missed changes are detected, the server enters lockdown mode, where only privileged clients can make requests. Any other server that is not missing changes can be used as a source for `dsreplication initialize`.

If a manual backup and restore of the server is required, then the following steps are equivalent to `dsreplication initialize`.

Performing a Manual Initialization

In the event that an online initialization is not possible, the following steps can be used to initialize a server.

1. From another server in the replication topology, backup the `userRoot`, `adminRoot`, `schema`, `replicationChanges` backends to the `/bak` directory. If encrypted attributes are present, then the `encryption-settings` backend should also be exported. One or more encryption settings IDs may need to be exported and imported.

```
$ <source-server-root>/bin/backup --backendID userRoot -d bak/userRoot
$ <source-server-root>/bin/backup --backendID adminRoot -d bak/adminRoot
$ <source-server-root>/bin/backup --backendID schema -d bak/schema
$ <source-server-root>/bin/backup --backendID replicationChanges -d bak/
replicationChanges
$ <source-server-root>/bin/encryption-settings export --id ID --output-file
 bak/exported-key
```

2. Copy the `bak` directory to the new replica.

```
$ scp -r <source-server-root>/bak <user>@<destination-server>:<destination-
server-root>/bak
```

3. Stop the server and restore the `userRoot`, `changelog`, `adminRoot`, `replicationChanges` backends. If the `encryption-settings` backend was exported, it should also be reimported.

```
$ <destination-server-root>/bin/restore -d bak/userRoot
$ <destination-server-root>/bin/restore -d bak/adminRoot
$ <destination-server-root>/bin/restore -d bak/schema
$ <destination-server-root>/bin/restore -d bak/replicationChanges
$ <destination-server-root>bin/encryption-settings import --input-file bak/
exported-key --set-preferred
     Enter the PIN used to encrypt the definition:
```

4. Start the server using `bin/start-server`.

Fixing Replication Conflicts

Replication conflicts can occur when an incompatible change to an entry is made on two replicas at the same time. The change is processed on one replica and then replicated to the other replica, which causes the conflict. While most conflicts are resolved automatically, some require manual action.

To fix replication conflicts, initialize the replica containing the conflicts with the data from another replica that does not have conflicts. If the database is large and the number of conflicts small, running `ldapmodify` against the server with the conflict will work if the command includes the Replication Repair Control specified by OID value 1.3.6.1.4.1.30221.1.5.2. The Replication Repair Control prevents the change from replicating. It also enables changing operational attribute values, which are not normally writable.

The following steps provide an example of using the Replication Repair Control to fix replication conflicts by applying change to only the server with the conflict. There are two examples: one for a modification conflict found by performing an `ldap-diff`, and the other for a naming conflict.

To Fix a Modify Conflict

1. The `bin/ldap-diff` tool can be used to isolate conflicting entries between two replicas. The following uses the tool to search across the entire base DN for any difference in user attributes, and reports the difference in `difference.ldif`. Replace the `sourceHost` value with the server that needs the adjustment.

```
$ bin/ldap-diff --sourceHost austin02.exmple.com --sourcePort 1389 \
                        --sourceBindDN "cn=Directory Manager" --
sourceBindPassword pass \
                        --targetHost austin01.example.com --targetPort 1389
 \
                        --targetBindDN "cn=Directory Manager" --
targetBindPassword
                        --baseDN "dc=example,dc=com" --outputLDIF
 difference.ldif \
                        --searchFilter "(objectclass=*)" --numPasses 3 "*"
 pass \
                        "^userPassword"
```

2. The `difference.ldif` file is in a format that can be used with `ldapmodify` to apply changes to the server that contains conflicts. The `ldap-diff` command must have been run with the `sourceHost` value as the server with conflicts. The following is an example of the contents of `difference.ldif`:

```
dn: uid=user.1,ou=people,dc=example,dc=com
                        changetype: modify
                        add: mobile
                        mobile: +1 568 232 6789
                        -
                        delete: mobile
                        mobile: +1 568 591 7372
                        -
```

3. Run `bin/ldapmodify` to correct the entries on only the server with conflicts.

```
$ bin/ldapmodify --bindPassword password -J "1.3.6.1.4.1.30221.1.5.2" \
                        --filename difference.ldif
```

To Fix a Naming Conflict

1. In this example, a naming conflict was encountered when the replica attempted to replay an ADD of `uid=user.200,ou=people,dc=example,dc=com`. In other words, two clients added the entry at the same time as an entry of the same name was added on another replica.

   ```
   [18/Feb/2010:14:53:12 -0600] category=EXTENSIONS severity=SEVERE_ERROR
   msgID=1880359005 msg="Administrative alert type=replication-unresolved-
   conflict
   id=bbd2cbaf-90a4-42af-94a8-c1a42df32fc6
   class=com.unboundid.directory.server.replication.plugin.ReplicationDomain
   msg='An unresolved conflict was detected for DN
    uid=user.200,ou=People,dc=example,dc=com.
   The conflicting entry has been renamed to
   entryuuid=69807e3d-ab27-43a3-8759-
   ec0d8d6b3107+uid=user.200,ou=People,dc=example,dc=com'"
   ```

2. The Directory Server prepends the entryUUID to the DN of the conflicting attribute and adds a `ds-sync-conflict-entry` auxiliary object class to the entry to aid in search. For example, the following command searches for any entry that has the `ds-sync-conflict-entry` objectclass and returns only the DNs that match the filter. You should see the conflicting entry for uid=user.200.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub \
     "(objectclass=ds-sync-conflict-entry)" "1.1"
   ```

   ```
   dn: entryuuid=69807e3d-ab27-43a3-8759-
   ec0d8d6b3107+uid=user.200,ou=People,dc=example,dc=com

   dn: entryuuid=523c430e-
   a870-4ebe-90f8-9cd811946420+uid=user.200,ou=People,dc=example,dc=com
   ```

   Note: Conflict entries are not returned unless the `objectclass=ds-sync-conflict-entry` is present in the search filter.

3. After comparing the conflict entry with the target entry, the difference can be applied in a manner similar to the previous example using `ldapmodify` with the Replication Repair Control. The conflict entry can also be deleted using this command. Run `bin/ldapmodify` with the Replication Repair Control to make the fix. When making changes using the Replication Repair Control, the updates will not be propagated via replication. You should examine each and every replica one by one, and apply the necessary modifications using the request control.

   ```
   $ bin/ldapmodify -J "1.3.6.1.4.1.30221.1.5.2" \
     --filename difference.ldif
   ```

Fixing Mismatched Generation IDs

A warning that multiple generation IDs were detected for a specific suffix indicates that one or more replicas need to be re-initialized. If the warning is presented from a server after an initialization, it could be that `post-external-initialization` was not run as part of a global change in data. Running this command will fix the situation. The `dsreplication` status command will warn when any generation IDs are different across the topology.

Replication Reference

The following section shows general reference information related to replication.

Summary of the dsreplication Subcommands

A summary of the `dsreplication` subcommands and functions is presented in the table below.

Table 48: dsreplication subcommands

Subcommand	Description

cleanup-local-server	Removes replication-related artifacts from the configuration, schema as well as the server registry while the local server is offline. The subcommand does not remove references to this server from other replicas and replication servers in the topology. Therefore, it is recommended to remove this server first from the replication topology either by using the disable or remove-defunct-server subcommands. Since this subcommand can only be executed when the server is offline, replication attributes from suffixes other than the server registry or the schema will not be removed. The tool will produce an LDIF file, `logs/cleanup-backends.ldif` that may be used to the remove replica state from the base entry of these suffixes after the server is restarted.
	To remove the replication history from regular suffixes, export the formerly replicated suffixes using the --excludeReplication option of the `export-ldif` command. The resulting LDIF file can be re-imported using the `import-ldif` command. For example:
	`$ bin/export-ldif --backendID userRoot --`
	`excludeReplication \`
	`  --ldifFile cleansed.ldif`
	`$ bin/import-ldif --backendID userRoot --ldifFile`
	` cleansed.ldif`
	Exporting using the --excludeReplication option of the `export-ldif` command will also remove the replica state from the output. The LDIF created by the cleanup-local-server subcommand does not need to be applied after the server is restarted.

disable	Disables replication on the specified server for the provided base DN and removes references to this server in the other servers with which it is replicating data.
enable	Updates the configuration of the servers to replicate the data under the specified base DN. If one of the servers is already replicating the data under the base DN with other servers, executing this subcommand will update the configuration of all the servers (so it is sufficient to execute the command line once for each server you add to the replication topology).
initialize	Initializes the data under the specified base DN on the destination server with the contents on the source server.
initialize-all	Initializes the data under the specified base DN on all servers in the replication topology with the contents on the specified server.
post-external-initialization	Used with `pre-external-initialization`, the command resets the generation ID based on the newly-loaded data. This subcommand must be called after initializing the contents of all the replicated servers using the `import-ldif` tool or `dsreplication initialize`. Specify the list of base DNs that have been initialized and provide the credentials of any of the servers that are being replicated. See the usage of the pre-external-initialization subcommand for more information. This subcommand only needs to be run on one of the replicas once.
pre-external-initialization	Clears the existing generation ID and removes all accumulated changes of the replicated suffix from the replication changelog database at each and every replication server. This subcommand should be used when globally restoring the replicas on all of the servers in a topology. You must specify the list of base DNs that will be initialized and provide the credentials of any of the servers that are being replicated. After calling this subcommand, initialize the contents of all the servers in the topology, then call the

Subcommand	Description
	post-external-initialization subcommand. This subcommand only needs to be run on one of the replicas once.
remove-defunct-server	Removes an offline defunct server from the replication on all servers in the topology.
status	Displays the status of replication domains. If no base DNs are specified as parameters, the information for all base DNs is displayed. Available options with the status subcommand are: --showAll, --displayServerTable, --location.

Summary of the Direct LDAP Monitor Information

The following table provides a description of the attributes in the `cn=Direct LDAP Server` monitor entry. The DN for the monitor entry is as follows:

```
dn: cn=Direct LDAP Server <baseDN> <host name:ldapPort> <serverID>,cn=monitor
```

Table 49: Direct LDAP Monitor Information

Monitor Attribute	Description
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this server is connected. The first number is the replication server port number and the second number is the server-id of the replication server.
replica-id:<serverID>	Replica ID number.
replication-backlog	Number of changes that the replication server has not seen from the server.
missing-changes	Number of missing changes.
approximate-delay	Difference between the time of the last change that the replication server has seen from the LDAP server and the most current time stamp on the latest change on the server.
base-dn	Base DN
ssl-encryption	Flag to indicate if SSL encryption is in use.
protocol-version	Displays the replication protocol version.
generation-id	Generation ID for the base DN on the Directory Server.
restricted	Boolean that indicates whether the replication domain is restricted in an Entry Balancing Configuration with the Directory Proxy Server.
ack-sent	Number of acknowledgement messages sent to this replica (not currently used).
ack-received	Number of acknowledgement messages received from this replica (not currently used).
add-sent	Number of protocol messages with an LDAP Add sent to this replica.
add-received	Number of protocol messages with an LDAP Add received from this replica.
delete-sent	Number of protocol messages with an LDAP Delete sent to this replica.
delete-received	Number of protocol messages with an LDAP Delete received from this replica.
done-sent	Number of done messages sent to this replica. A done message indicates an end of online initialization session. If the value is non-zero, then this replica has been initialized over the replication protocol.

Monitor Attribute	Description
done-received	Number of done messages received from this replica. A done message indicates an end of online initialization session. If the value is non-zero, then this replica has completed the initialization of other replicas over the replication protocol.
entry-sent	Number of entry messages sent to this replica. Entry messages carry replicated data to initialize replicas over the replication protocol.
entry-received	Number of entry messages received from this replica. Entry messages carry replicated data to initialize replicas over the replication protocol.
error-sent	Number of error messages sent to this replica.
error-received	Number of error messages received from this replica.
heartbeat-sent	Number of heartbeat messages sent to this replica.
heartbeat-received	Number of heartbeat messages received from this replica (should always be 0, since the replica never sends a heartbeat message to the server).
initialize-request-sent	Number of initialize-request messages sent to this replica. This message is sent when another replica requested initialization of its data using the replication protocol.
initialize-request-received	Number of initialize-request messages received from this replica. This message is sent when this replica requested initialization of its data using the replication protocol from another replica.
initialize-target-sent	Number of initialize-target messages sent to this replica. This message is sent before another replica has started the initialization of this replica.
initialize-target-received	Number of initialize-target messages received from this replica. This message is sent before this replica starts the initialization of one or more replicas.
modify-sent	Number of protocol messages with an LDAP Modify sent to this replica.
modify-received	Number of protocol messages with an LDAP Modify received from this replica.
modify-dn-sent	Number of protocol messages with an LDAP Modify DN sent to this replica.
modify-dn-received	Number of protocol messages with an LDAP Modify DN received from this replica.
repl-server-start-sent	Number of replication-server-start messages sent to this replica (should never be more than 1). The Replication Server responds with this message to the start message received from the replica.
repl-server-start-received	Number of replication-server-start messages received from this replica (should always be 0).
reset-generation-id-sent	Number of reset generation ID messages received from this replica.
reset-generation-id-received	Number of reset generation ID messages sent to this replica (should always be 0).
server-start-sent	Number of server-start messages sent to this replica (should always be 0).
server-start-received	Number of server-start messages received from this replica (should never be more than 1). Server-start is the first message the replica sends after establishing a replication connection.

Monitor Attribute	Description
window-sent	Number of window messages sent to this replica. Window messages are used for cumulative acknowledgement in the replication protocol.
window-received	Number of window messages received from this replica. Window messages are used for cumulative acknowledgement in the replication protocol.
window-probe-sent	Number of window probe messages sent to this replica (should always be 0).
window-probe-received	Number of window probe messages received from this replica. The replica sends a window probe message to the server if the send window in the replica is closed and the replica is unable to publish updates to the server.
update-sent	Number of changes sent to this server.
update-received	Number of changes received from this server.
internal-connection	Indicates if the replica is in the same process as the replication server.
server-state	Displays the state of the replica. Displays the latest change number that the replica has seen from all the other replicas including itself.
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
max-send-window	Maximum number of changes that can be sent to the LDAP server before requiring an ACK.
current-send-window	Current number of changes remaining to be sent to the LDAP server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received by the LDAP server before sending an ACK.
current-rcv-window	Number of changes remaining to be received by the LDAP server before sending an ACK Server.
degraded	Indicates that the generation ID of the replica does not match the generation ID of the server. This is a temporary state, when loading data into the topology or the replica has not been initialized. Normally, it should be false.

Summary of the Indirect LDAP Server Monitor Information

The following table provides a description of the attributes in the `cn=Indirect LDAP Server` monitor entry. These attributes provide information about a Directory Server that is connected to a different replication server in the topology.

```
dn: cn=Indirect LDAP Server <baseDN> <serverID>,cn=monitor
```

Table 50: Indirect LDAP Server Monitor Information

Monitor Attribute	Description
replica-id: <serverID>	ID number identifying the replica.
base-dn: <baseDN>	Base DN
connected-to: Remote Repl Server <baseDN> <host name:replPort> <replID>	Replication server to which the directory server is connected.
replication-backlog	Number of changes that the replication server has not seen from the server.
approximate-delay	Amount of time between the last change seen by this Directory Server and the most recent change seen by the remote replication server. This value is the amount of time between the time stamps, not the amount of time required to synchronize the two servers.
generation-id	Generation ID for this suffix on this remote replication server.
consumed-update-recent-rate	Rate that the connected Replication Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Replication Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.

Summary of the Remote Replication Server Monitor Information

The following table provides a description of the attributes in the `cn=Remote Repl Server` monitor entry. The DN for the monitor entry is as follows:

```
dn: cn=Remote Repl Server <baseDN> <host name:replPort> <serverID>,cn=monitor
```

Table 51: Remote Replication Server Monitor Information

Monitor Attribute	Description
replication-server: <host name>:<repl port>	Host name and replication port number of the Replication Server.
replication-server-id:<serverID>	Server ID for the Replication Server.
available	Indicates if the remote replication server is available or not. Values: true or false.
sending-paused	Indicates if sending is paused. Values: true or false.
receiving-paused	Indicates if receiving is paused. Values: true or false.
wan-gateway-priority	Specifies the WAN Gateway priority of the remote replication server.
is-wan-gateway	Indicates if the remote replication server is a WAN Gateway. Values: true or false.
wan-gateway-desired	Indicates if the remote replication server is a desired gateway. Values: true or false. This entry together with the `is-wan-gateway` property indicates the desired state. **is-wan-gateway=false, wan-gateway-desire=false**: Indicates another server with a higher gateway priority exists or the gateway priority is set to disabled.

Monitor Attribute	Description
	is-wan-gateway=false, wan-gateway-desire=true: Indicates that the remote replication server wants to be a WAN gateway. **is-wan-gateway=true, wan-gateway-desire=false**: Indicates that the remote replication server wants to give up its role as a WAN gateway. **is-wan-gateway=false, wan-gateway-desire=true**: Indicates the remote replication server wants to remain as a gateway server.
base-dn	base DN
ssl-encryption	Flag to indicate if SSL encryption is in use.
protocol-version	Replication protocol version.
generation-id	Generation ID for this suffix on this remote replication server.
restricted	Indicates that remote replication server is in an entry-balancing deployment.
add-sent	Number of protocol messages with an LDAP Add sent to the remote replication server.
add-received	Number of protocol messages with an LDAP Add received from the remote replication server.
delete-sent	Number of protocol messages with an LDAP Delete sent to the remote replication server.
delete-received	Number of protocol messages with an LDAP Delete received from the remote replication server.
done-sent	Number of done messages sent to the remote replication server. A done message indicates an end of online initialization session.
done-received	Number of done messages received from the remote replication server. A done message indicates an end of online initialization session.
entry-sent	Number of entry messages sent to the remote replication server. Entry messages carry replicated data to initialize replicas over the replication protocol.
entry-received	Number of entry messages received from the remote replication server. Entry messages carry replicated data to initialize replicas over the replication protocol.
error-sent	Number of error messages sent to the remote replication server.
error-received	Number of error messages received from the remote replication server.
heartbeat-sent	Number of heartbeat messages sent to the remote replication server.
heartbeat-received	Number of heartbeat messages received from the remote replication server.
initialize-request-sent	Number of initialize-request messages sent to the remote replication replication server. This message is used during online initialization from one replica to another.
initialize-request-received	Number of initialize-request messages received from the remote replication server. This message is used during online initialization from one replica to another.
initialize-target-sent	Number of initialize-target messages sent to the remote replication server. This message is used before online initialization from one replica to another.

Monitor Attribute	Description
initialize-target-received	Number of initialize-target messages received from the remote replication server. This message is used before online initialization from one replica to another.
modify-sent	Number of protocol messages with an LDAP Modify sent to the remote replication server.
modify-received	Number of protocol messages with an LDAP Modify received from the remote replication server.
modify-dn-sent	Number of protocol messages with an LDAP Modify DN sent to the remote replication server.
modify-dn-received	Number of protocol messages with an LDAP Modify DN received from the remote replication server.
monitor-sent	Number of monitor messages sent to the remote replication server. This message is primarily used when communicating with directory servers running a prior release.
monitor-received	Number of monitor messages received from the remote replication server. This message is primarily used when communicating with directory servers running a prior release.
monitor-request-sent	Number of monitor requests sent to the remote replication server. The receiving server will respond with a monitor message that includes server's information about the state of the topology.
monitor-request-received	Number of monitor requests received from the remote replication server. This server will respond with a monitor message that includes server's information about the state of the topology.
monitor-v2-sent	Number of monitor messages sent to the remote server. This monitor message is only used with Directory Server v3.5 or later.
monitor-v2-received	Number of monitor messages received from the remote server. This monitor message is only used with Directory Server v3.5 or later.
pause-sending-updates-sent	Number of pause-sending-updates messages sent to the remote server. The remote server must stop sending update messages to this server when receiving this message.
pause-sending-updates-received	Number of pause-sending-updates messages received from the remote server. This server will stop sending update messages to the remote server upon receiving this message.
repl-server-start-sent	Number of replication-server-start messages sent to the remote replication server (should never be more than 1). The Replication Server responds with this message to the replication-server-start message received from remote replication servers.
repl-server-start-received	Number of replication-server-start messages received from the remote replication server (should never be more than 1) window-sent: the number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
reset-generation-id-sent	Number of reset generation ID messages received from the remote replication server. This message is sent before and after the data is initialized in the topology.

Monitor Attribute	Description
reset-generation-id-received	Number of reset generation ID messages sent to the remote replication server. This message is sent before and after the data is initialized in the topology.
server-info-sent	Number of replication server information messages sent to the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
server-info-received	Number of replication server information messages received from the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
set-source-location-sent	Number of set-source-locations messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
set-source-location-received	Number of set-source-locations messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
start-sending-updates-sent	Number of start-sending-updates messages sent to the remote replication server. The remote server may only start sending updates to this server after receiving this message.
start-sending-updates-received	Number of start-sending-updates messages received from the remote server. Sending update messages to the remote server may only start after receiving this message.
window-sent	Number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
window-received	Number of window messages received from the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
messages-sent	Total number of messages sent.
messages-received	Total number of messages received.
update-sent	Total number of updates sent.
update-received	Total number of updates received.
server-state	Displays the server state of the remote replication server. It displays the last change seen on the remote replication server.
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.

Monitor Attribute	Description
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
max-send-window	Maximum number of changes that can be sent to the remote replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the remote replication server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received from the remote replication server before sending an ACK.
current-rcv-window	Number of changes remaining to be received from the remote replication server before sending an ACK.
degraded	Indicates that the generation ID of the replica does not match the generation ID of the remote replication server. This is a temporary state, when loading data into the topology or the replica has not been initialized. Normally, it should be false.

Summary of the Replica Monitor Information

The following table provides a description of the attributes in the `cn=Replica` monitor entry for a specific base DN.

```
dn: cn=Replica <baseDN>,cn=monitor
```

Table 52: Indirect LDAP Server Monitor Information

Monitor Attribute	Description
base-DN: <baseDN>	Specified base DN. The monitor entries track your company's base DN (or `dc=example,dc=com`), `cn=schema`, and `cn=topology,cn=config`.
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this LDAP Server is connected. The first number is the replication server port number and the second number is the `server-id` of the replication server.
lost-connections	Number of times the Directory Server has lost connection to a replication server.
received-updates	Number of updates that the Directory Server Replica has received from the connected replication server.
sent-updates	Number of updates sent to the replication server.
pending-updates	Number of updates pending to send to the replication server.
replayed-updates	Total number of updates from the replication server that have been replayed for this replica.
replayed-updates-ok	Number of updates for this replica that have been successfully replayed with no conflicts.
replayed-update-failed	Number of updates for this replica that were successfully replayed after automatically resolving a modify conflict.
resolved-modify-conflicts	Number of updates for this replica that were successfully resolved after a modify conflict.

Monitor Attribute	Description
resolved-naming-conflicts	Number of updates for this replica that were successfully resolved after a naming conflict.
unresolved-naming-conflicts	Number of updates for this replica that could not be replayed due to an unresolvable naming conflict.
replica-id	Server ID for this replica.
max-rcv-window	Maximum number of changes that the Directory Server Replica can receive at a time before sending an acknowledgment back to the replication server.
current-rcv-window	Current received window size for this replica.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-rcv-window	Number of changes remaining to be received from the replication server before it must send an ACK.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the replication server before requiring an ACK.
ssl-encryption	Flag to indicate if SSL encryption is in use.
generation-id	Generation ID for this suffix on the Directory Server.
replication-backlog	Number of changes that are from this replica.

Summary of the Replication Server Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server` monitor entry.

```
dn: cn=Replication Server <baseDN> <replServerID>,cn=monitor
```

Table 53: Replication Server Monitor Information

Monitor Attribute	Description
replication-server-id	Server ID for the Replication server ID.
replication-server-port	Port number on which the replication server listens for communication from other servers.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
Generation IDs by Base DN	List of generation IDs for each base DN on the server.
num-outgoing-replication-server-connections	Number of outgoing connections from the replication server.
num-incoming-replication-server-connections	Number of incoming connections into the replication server.
num-incoming-replica-connections	Number of incoming connections to the replica.

Summary of the Replication Server Database Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server database` monitor entry.

```
dn: cn=Replication Server database <baseDN> <replServerID>,cn=monitor
```

Table 54: Replication Server Database Monitor Information

Monitor Attribute	Description
database-replica-id: <replicaID>	Specifies the replication server ID.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
first-change	First change number that is in this replication database table for this suffix from this `server-id`. For example, an example entry looks like the following: `0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011`
last-change	Last change number that is in this replication database table for this suffix from this `server-id`. For example, an example entry looks like the following: `0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011`
queue-size	Number of changes in the replication server queue waiting to be sent to this remote replication server database.
queue-size-bytes	Size in bytes of all the messages waiting in the queue.
records-added	Displays the number of records changed or added to the DIT.
records-removed	Displays the number of records removed from the DIT.

Summary of the Replication Server Database Environment Monitor Information

Content-Reference to:

The following table provides a description of the attributes in the `cn=Replication Server Database Environment` monitor entry, which includes the environment variables associated with the Oracle Berkeley Database Java Edition backend.

```
dn: cn=Replication Server Database Environment,cn=monitor
```

Table 55: Replication Server Database Environment Monitor Information

Monitor Attribute	Description
je-version	Current version of the Oracle Berkeley Java Edition.
current-db-cache-size	Current DB cache size.
max-db-cache-size	Maximum DB cache size.
db-cache-percent-full	Percentage of the cache used by the Directory Server.
db-directory	Directory that holds the changelogDb file.
db-on-disk-size	Size of the DB on disk.
cleaner-backlog	Number of log files that must be cleaned for the cleaner to meet its target utilization.
random-read-count	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.

Monitor Attribute	Description
random-write-count	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
sequential-read-count	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
sequential-write-count	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
nodes-evicted	Accumulated number of nodes evicted.
active-transaction-count	Number of currently active transactions.
num-checkpoints	Number of checkpoints. A checkpoint is a process that writes to your log files all the internal BTree nodes and structures modified as a part of write operations to your log files to facilitate a quick recovery.
checkpoint-in-progress: false	Indicates if a checkpoint is in progress.
total-checkpoint-duration-millis	Total time in milliseconds for all checkpoints.
average-checkpoint-duration-millis	Average time in milliseconds for all checkpoints.
last-checkpoint-duration-millis	Duration in milliseconds of the last checkpoint run.
last-checkpoint-start-time	Start time of the last checkpoint.
last-checkpoint-stop-time	Stop time of the last checkpoint.
millis-since-last-checkpoint	Time in milliseconds since the last checkpoint.
read-locks-held	Total read locks currently held.
write-locks-held	Total write locks currently held.
transactions-waiting-on-locks	Total transactions waiting for locks
je-env-stat-AdminBytes	Number of bytes of JE cache used for log cleaning metadata and other administrative structure.
je-env-stat-BufferBytes	Total memory currently consumed by log buffers, in bytes.
je-env-stat-CacheDataBytes	Total memory of cache used for data.
je-env-stat-CacheTotalBytes	Total amount of JE cache in use, in bytes.
je-env-stat-CleanerBacklog	Number of files to be cleaned to reach the target utilization.
je-env-stat-CursorsBins	Number of bottom internal nodes (BINs) encountered by the INCompressor that had cursors referring to them when the compressor ran. The compressor thread cleans up the internal BTree as records are deleted to ensure unused nodes are not present.
je-env-stat-DataBytes	Amount of JE cache used for holding data, keys and internal Btree nodes, in bytes.
je-env-stat-DbClosedBins	Number of bins encountered by the INCompressor that had their database closed between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-EndOfLog	Location of the next entry to be written to the log.
je-env-stat-InCompQueueSize	Number of entries in the INCompressor queue when the getStats() call was made.
je-env-stat-LastCheckpointEnd	Location in the log of the last checkpoint end.

Monitor Attribute	Description
je-env-stat-LastCheckpointId	ID of the last checkpoint.
je-env-stat-lastCheckpointStart	Location in the log of the last checkpoint start.
je-env-stat-lockBytes	Number of bytes of JE cache used for holding locks and transactions.
je-env-stat-NBINsStripped	Number of BINS stripped by the evictor.
je-env-stat-NCacheMiss	Total number of requests for database objects which were not in memory.
je-env-stat-NCheckpoints	Total number of checkpoints run so far.
je-env-stat-NCleanerDeletions	Number of cleaner file deletions this session.
je-env-stat-NCleanerEntriesRead	Accumulated number of log entries read by the cleaner.
je-env-stat-NCleanerRuns	Number of cleaner runs this session.
je-env-stat-NClusterLNsProcessed	Accumulated number of leaf nodes (LNs) processed because they qualify for clustering.
je-env-stat-NDeltaINFlush	Accumulated number of delta internal nodes (INs) flushed to the log.
je-env-stat-NEvictPasses	Number of passes made to the evictor.
je-env-stat-NFSyncRequests	Number of fsyncs requested through the group commit manager. `Fsync()` synchronizes the filesystem after a write or a transaction.
je-env-stat-NFSyncTimeouts	Number of fsync requests submitted to the group commit manager which timed out.
je-env-stat-NFSyncs	Number of fsyncs issued through the group commit manager.
je-env-stat-NFileOpens	Number of times a log file has been opened.
je-env-stat-NFullBINFlush	Accumulated number of full bottom internal nodes (BINS) flushed to the log.
je-env-stat-NFullINFlush	Accumulated number of full INs flushed to the log.
je-env-stat-NINsCleaned	Accumulated number of INs cleaned.
je-env-stat-NINsDead	Accumulated number of INs that were not found in the tree anymore (deleted).
je-env-stat-NINsMigrated	Accumulated number of INs migrated.
je-env-stat-NINsObsolete	Accumulated number of INs obsolete.
je-env-stat-NLNQueueHits	Accumulated number of LNs processed without a tree lookup.
je-env-stat-NLNsCleaned	Accumulated number of LNs cleaned.
je-env-stat-NLNsDead	Accumulated number of LNs that were not found in the tree anymore (deleted).
je-env-stat-NLNsLocked	Accumulated number of LNs encountered that were locked.
je-env-stat-NLNsMarked	Accumulated number of LNs that were marked for migration during cleaning.
je-env-stat-NLNsMigrated	Accumulated number of LNs migrated.
je-env-stat-NLNsObsolete	Accumulated number of LNs obsolete.
je-env-stat-NLogBuffers	Number of log buffers currently instantiated.
je-env-stat-NMarkedLNsProcessed	Accumulated number of LNs processed because they were previously marked for migration.

Monitor Attribute	Description
je-env-stat-NNodesExplicitlyEvicted	Accumulated number of nodes evicted.
je-env-stat-NNodesScanned	Accumulated number of nodes scanned to select the eviction set.
je-env-stat-NNodesSelected	Accumulated number of nodes selected to evict.
je-env-stat-NNotResident	Number of requests for database objects not contained within the in memory data structures.
je-env-stat-NOpenFiles	Number of files currently open in the file cache.
je-env-stat-NPendingLNsLocked	Sccumulated number of pending LNs that could not be locked for migration because of a long duration application lock.
je-env-stat-NPendingLNsProcessed	Accumulated number of LNs processed because they were previously locked.
je-env-stat-NRandomReadBytes	Number of bytes read which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomReads	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWriteBytes	Number of bytes written which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWrites	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NRepeatFaultReads	Number of reads which had to be repeated when faulting in an object from disk because the read chunk size controlled by `je.log.faultReadSize` is too small.
je-env-stat-NRepeatIteratorReads	Number of times we try to read a log entry larger than the read buffer size and can't grow the log buffer to accommodate the large object.
je-env-stat-NRootNodesEvicted	Accumulated number of database root nodes evicted.
je-env-stat-NSequentialReadBytes	Number of bytes read which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialReads	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWriteBytes	Number of bytes written which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWrites	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NSharedCacheEnvironments	Number of environments using the shared cache.
je-env-stat-NTempBufferWrites	Number of writes which had to be completed using the temporary marshalling buffer because the fixed size log buffers specified by `je.log.totalBufferBytes` and `je.log.numBuffers` were not large enough.
je-env-stat-NToBe-CleanedLNsProcessed	Accumulated number of LNs processed because they are soon to be cleaned.
je-env-stat-NonEmptyBins	Number of non-empty bins.
je-env-stat-ProcessedBins	Number of bins that were successfully processed by the INCompressor.

Monitor Attribute	Description
je-env-stat-RequiredEvictBytes	Number of bytes that must be evicted to get within the memory budget.
je-env-stat-SharedCacheTotalBytes	Total amount of the shared JE cache in use, in bytes.
je-env-stat-SplitBins	Number of bins encountered by the INCompressor that were split between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-TotalLogSize	Approximation of the current total log size in bytes.
je-env-stat-NOwners	Total lock owners in the lock table.
je-env-stat-NReadLocks	Total read locks currently held.
je-env-stat-NRequests	Total number of lock requests to date.
je-env-stat-NTotalLocks	Total locks currently in the lock table.
je-env-stat-NWaiters	Total transactions waiting for locks.
je-env-stat-NWaits	Total number of lock waits to date.
je-env-stat-NWriteLocks	Total write locks currently held.
je-env-stat-LastCheckpointTime	Time of the last checkpoint.
je-env-stat-LastTxnId	Last transaction ID allocated.
je-env-stat-NAborts	Number of transactions that have aborted.
je-env-stat-NActive	Number of transactions that are currently active.
je-env-stat-NBegins	Number of transactions that have begun.
je-env-stat-NCommits	Number of transactions that have committed.
je-env-stat-NXAAborts	Number of XA transactions that have aborted.
je-env-stat-NXACommits	Number of XA transactions that have committed.
je-env-stat-NXAPrepares	Number of XA transactions that have been prepared.

Summary of the Replication Summary Monitor Information

The following table provides a description of the attributes in the `cn=Replication Summary` monitor entry.

```
dn: cn=Replication Summary <baseDN>,cn=monitor
```

Table 56: Replication Summary Monitor Information

Monitor Attribute	Description
base-dn:<baseDN>	Base DN summary.
replica: replica-id, ldap-server connected-to, generation-id, replication-backlog, recent-update-rate, peak-update-rate, age-of-oldest-backlog-change	Summary information for each replica in the topology. This entry appears for each replica in the topology with its own respective properties.
replication-server: server-id, server, generation-id, status, last-connected, last-failed, failed-attempts, attributes.	Summary information for each remote replication server only in the topology. This entry appears for each Replication Server in the topology with its own respective `serverID` and `server` properties.
update-queue: id, max-count, current-count, max-size, current-	Summary information for each update queue on a server:

Monitor Attribute	Description
size, polling-source, polling-source-changed	**id**. The ID of the receiving replica or replication server **max-count**. The maximum number of update messages that the sending replication server will keep in memory for the receiving replica or replication server. If the receiver cannot accept messages fast enough for any reason (high load, network latency, etc), then this queue will fill up. When that happens, the sending replication server will read update messages from the changelog backend. This slows down the update processing considerably. **current-count**. The number of update messages currently on the queue that have not been sent to the receiving replica or replication server. Every time the sending replication server sends an update to the receiving replica or replication server, this counter is decremented. **max-size**. The maximum total size (in bytes) of update messages that may be in the queue. This queue is capped by both the maximum count (`max-count`) and the `max-size` setting, whichever is reached first. **current-size**. The total size of update messages currently on the queue that have not been sent to the receiving replica or replication server. Every time the sending replication server sends an update to the receiving replica or replication server, this value is decremented by the size of the published update message. **polling-source**. Either 'memory' or 'db'. If set to 'memory', the sending replication server relies only on the in-memory queue to push update messages to the receiving replica or replication server. If set to 'db', update messages are read and sorted from the changelog database, which is significantly slower than publishing updates from the in-memory queue. **polling-source-changed**. The total number of times the polling-source attribute has changed value (either from 'db' to 'memory' or from 'memory' to 'db'). If this value changes very frequently, then the queue size setting is probably too low.

Summary of the replicationChanges Backend Monitor Information

The following table provides a description of the attributes in the `cn=replicationChanges Backend` monitor entry.

```
dn: cn=replicationChanges Backend,cn=monitor
```

Table 57: replicationChanges Backend Monitor Information

Monitor Attribute	Description
ds-backend-id	ID descriptor for the backend. Typically, this will be "replicationChanges".
ds-backend-base-dn	Base DN for the backend. Typically, this will be `cn=replicationChanges`.
ds-backend-is-private	Flag to indicate if the backend is private.
ds-backend-entry-count	Entry count for the backend.
ds-base-dn-entry-count	Entry count for the base DN and the specified base DN.
ds-backend-writability-mode	Flag to indicate if the backend is writable or not.

Summary of the Replication Protocol Buffer Monitor Information

The following table provides a description of the attributes in the `cn=Replication Protocol` Buffer monitor entry. The monitors provide information on the state of the buffer for protocol operations, which is kept in the local storage.

```
dn: cn=Replication Protocol Buffer,cn=monitor
```

Table 58: Replication Protocol Buffer Monitor Information

Monitor Attribute	Description
saved-buffers	Number of protocol buffers. Initial buffer size is 4k.
reallocations	Number of times the buffers had to be reallocated due to insufficient size.
large-buffer-creates	Number of times a buffer larger than 512k was requested.
large-buffer-evictions	Number of times a buffer was removed from the thread local storage, because it has grown above 512k.

Advanced Topics Reference

This chapter presents background reference information covering advanced replication topics.

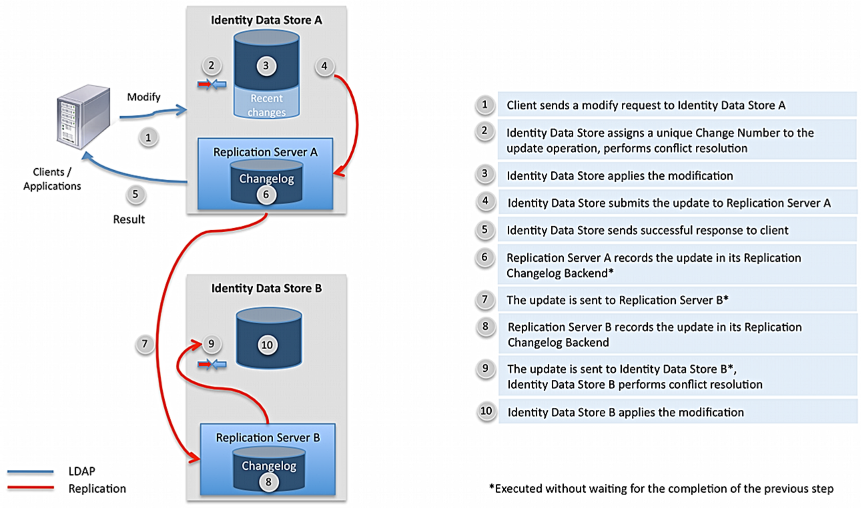
About the Replication Protocol

Replication communicates using a proprietary binary protocol that is implemented on top of the TCP/IP protocol using SSL encryption. Some protocol messages are used for administrative purposes (such as WAN Gateway server negotiation or flow control), some carry updates to replicated data, while others are directed to all servers for monitoring requests.

In a replicated topology, each participating Directory Server is connected to every other server via the replication server port in order to monitor health. Servers which share the same location setting are also connected to rapidly replicate changes and lastly the WAN Gateway servers are all interconnected to replicate changes across locations.

Directory Servers keep connections open as long as possible to reduce the communication latency when messages are exchanged. Heartbeat messages are transmitted on a regular basis to detect a network failure or an unresponsive directory server as early as possible. Heartbeat messages also prevent idle connections from being closed by firewalls.

The following detailed communication flow will be used to describe major components of replication. This illustration is the expanded view of figure shown in the Overview section.



1 Client sends a modify request to Identity Data Store A

2 Identity Data Store assigns a unique Change Number to the update operation, performs conflict resolution

3 Identity Data Store applies the modification

4 Identity Data Store submits the update to Replication Server A

5 Identity Data Store sends successful response to client

6 Replication Server A records the update in its Replication Changelog Backend*

7 The update is sent to Replication Server B*

8 Replication Server B records the update in its Replication Changelog Backend

9 The update is sent to Identity Data Store B*, Identity Data Store B performs conflict resolution

10 Identity Data Store B applies the modification

*Executed without waiting for the completion of the previous step

Figure 15: Replication Communication Flow

Step 1. Client sends a Modify request to Directory Server A.

Step 2. Directory Server A assigns a unique change number to the operation. Conflict resolution is executed to see if the Modify request is in conflict with the existing attribute types or values in the existing entry. The change number is assigned before the Directory Server backend is updated so that the arrival order of client requests can be preserved. Historical data in the target entry is updated to reflect the change. Note that historical data is only updated for ADD and MODIFY operations.

Step 3. Directory Server applies the modifications in the corresponding backend.

Step 4. If the MODIFY operation successfully completes, then the Directory Server will submit the update to its embedded *Replication Server*. The Replication Server is a component within the Directory Server process responsible for propagating updates to and from the replication topology. The Directory Server itself only communicates with a single replication server, whereas the replication server component is connected to all other replication servers. If the Directory Server process exits unexpectedly and some updates have not been passed to the Replication Server, the backend has the ability to recover the last 50,000 recent changes that were processed at this server, guaranteeing that these changes can be replicated when the server starts up. The figure above also shows that replication protocol is used not just between replication servers but also between the Directory Server and the Replication Server.

Step 5. The response is sent to the client. In this example, a successful response is assumed.

Step 6. The Replication Server records the update in its own Changelog backend (i.e., backend ID of `replicationChanges`) and on disk with the path to `changelogDb` under the server root. The Replication Changelog backend keeps track of updates at each and every Directory Server in the replication topology. When a Directory Server joins the replication topology after being disconnected for some reason, updates from the Replication Changelog backend are re-sent to this Directory Server. Old records from the Replication Changelog backend are purged, which by default removes records older than 24 hours. If the backend does not contain all of the records that another Directory Server needed when rejoining the replication topology, then the replicated data set in the Directory Server must be re-initialized. In this case, the Directory Server enters lockdown mode and an administrative alert is sent.

Step 7. The Replication Server submits the update to the replication server component in Directory Server B. If there were more Directory Servers in this example, the Replication Server would submit the update to all the other replication servers in the same location.

Step 8. Just like in Step 6, the Replication Server component receiving an update inserts the change into its Replication Changelog backend.

Step 9. The update is forwarded to the Replica in Directory Server B. Conflict resolution is executed to see if the Modify request is in conflict with the existing attribute types or values in the existing entry.

Step 10. The Directory Server applies the modification in the corresponding backend. The Recent Changes database is not updated, because only updates that originated at this Directory Server are recorded in the Recent Changes database.

Change Number

As seen in the previous Figure, the Directory Server assigns a unique change number to each update operation (specifically, ADD, DELETE, MODIFY, or MODIFY DN operations) to track each request when received from a client. The change number not only identifies each and every update, but it also allows ordering updates to the replicated data the same way on each Directory Server. The change number is composed of the following multiple fields:

- **Timestamp** that identifies when the update was made. The timestamp is time-zone-independent and recorded with millisecond resolution.
- **Server ID** that uniquely identifies the Directory Server where the update was made.
- **Sequence number** that defines the order of the updates received from external clients at a particular directory server.

The replication protocol also sets a virtual clock that eliminates the need for synchronized time on servers. For troubleshooting purposes, however, it is still recommended to keep the system clocks synchronized.

Conflict Resolution

The eventual-consistency model employed in replication introduces a window where conflicting updates targeting the same entry may be applied at two different Directory Servers. In general, two updates to the same Directory Server are in conflict if the update that arrived later fails. Conflict resolution, when possible, corrects conflicts introduced by clients automatically. There are some exceptions, however, when manual administrative action is required. For example, adding an entry in one replica and deleting the parent of this entry on another replica simultaneously will introduce a conflict that requires manual action. In a carefully implemented deployment, the risk of introducing conflicts that require manual action can be significantly reduced or even eliminated.

The conflict resolution algorithm in the PingDirectory Server uses a mechanism that orders all updates in the replication topology. Each update in the Directory Server is assigned a unique change number. The change number is attached to each update propagated via replication and allows each Directory Server to order updates exactly the same way.

Consider the following example that results in a conflict: add a single-valued attribute with different values to an entry concurrently at two Directory Servers (shown in the figure below). It is easy to see that the second operation would fail if a client attempted to add the same attribute to the same entry at the same Directory Server. In a replicated environment, the conflict is not immediately seen if these updates are applied concurrently at two different Directory Servers. The conflict is handled only after replication propagates the updates. The Directory Servers resolve the conflict independently of the other server. On one Directory Server, the entry will be updated to reflect the correct value; on the other Directory Server, the value will stay the same. As result, each Directory Server will independently resolve the conflict the same way based on the ordering of the updates. This example is illustrated below:

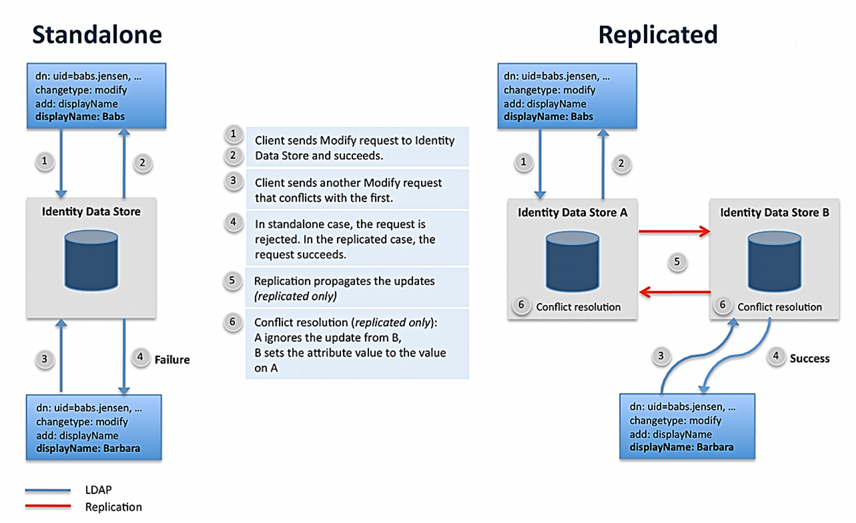


Figure 16: Conflict Resolution Process Flow

WAN-Friendly Replication

Many multi-national corporations that have data centers in different countries must minimize latency over WAN to ensure acceptable performance for their client applications. To minimize WAN latency, the Directory Server assigns one of two roles to the replication servers: the role of standard replication transmitting updates to the other co-located replication servers; the other role, a WAN-dedicated replication server designed to send updates to other WAN-designated replication servers in other locations. This two-role system minimizes WAN traffic by pushing all replication updates onto the connected replication servers that are designated as WAN Gateway Servers. Only the designated WAN Gateway Servers can transmit the update messages to other connected WAN Gateway servers at other locations.

WAN Gateway Server

The Directory Server's replication mechanism relies on the server's location information to reduce protocol traffic on WAN links. During protocol negotiation, the replication server with the highest WAN Gateway priority (priority 1 indicates the highest priority) automatically assumes the role as the WAN Gateway Server for that particular location. The Gateway Server's main function is to route update messages from other non-gateway servers at the same location to remote WAN Gateway servers at other locations. Similarly, at the destination point, the replication server with the WAN Gateway role will receive update messages from other WAN gateway servers at other locations and push them out to all replication servers at the current location. This setup ensures that all WAN communication flows through the WAN Gateway Servers.

The figure below shows a basic connection configuration for updates. Keep in mind that all of the replication servers are fully connected to each other for monitoring or server negotiation purposes.

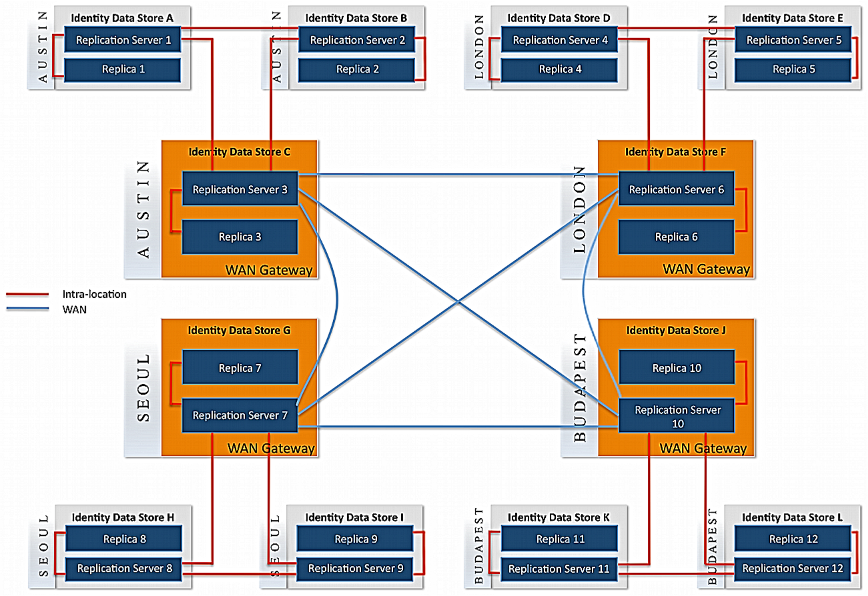


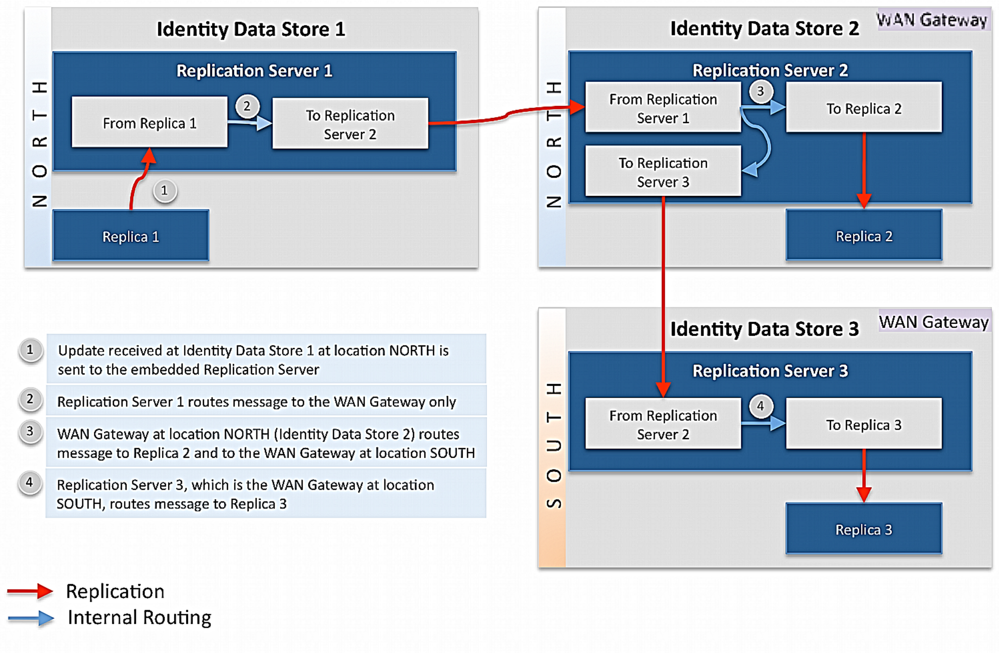
Figure 17: WAN Gateway Servers

If the WAN Gateway Server is temporarily unavailable due to a planned or unplanned downtime, the system will dynamically re-route updates to a newly designated WAN Gateway Server in the same location. The replication server with the next highest WAN Gateway priority number automatically assumes the WAN Gateway role. For deployments with entry-balancing Directory Proxy Servers, there will be one WAN Gateway Server per data set.

By default, all servers are enabled to serve as WAN Gateways and all are set to priority 5, which is simply a way to make them all equal. If necessary, the WAN Gateway priority can be changed using `dsconfig` after replication has been enabled.

WAN Message Routing

Non-gateway replication servers forward update messages from replicas to co-located replication servers only. It is the responsibility of the WAN gateway server to forward these messages to WAN gateway servers at other locations. The figure below illustrates how an update message is routed from a non-gateway server to a remote location.



1 Update received at Identity Data Store 1 at location NORTH is sent to the embedded Replication Server

2 Replication Server 1 routes message to the WAN Gateway only

3 WAN Gateway at location NORTH (Identity Data Store 2) routes message to Replica 2 and to the WAN Gateway at location SOUTH

4 Replication Server 3, which is the WAN Gateway at location SOUTH, routes message to Replica 3

Figure 18: WAN Message Routing

WAN Gateway Server Selection

The WAN Gateway role is dynamically assigned to the most suitable server in a particular location. In most cases, the replication server with the higher WAN Gateway priority (e.g., priority 1 indicates the highest priority) assumes this role.

A replication server will not attempt to become a WAN Gateway if any of the following conditions exists:

The WAN Gateway priority is set to 0

The replication server is backlogged at server startup

The current WAN Gateway has higher priority

The WAN Gateway has not been elected yet, but higher priority replication servers are present in the location

The currently active WAN Gateway server will give up the gateway role in the following cases:

The server has started the shutdown process

The server is preparing for a scheduled maintenance cycle

The server learns about a higher priority replication server in its location

Replication servers send WAN Gateway information to other servers at regular intervals using the replication protocol. This allows the replication servers to take the appropriate action, for example, to become a WAN Gateway, if necessary without any manual administrative action.

WAN Replication in Mixed-Version Environments

WAN Gateway Role assignment is only available on PingDirectory Servers version 3.5 or later. Legacy servers (i.e., pre-3.5) will never be selected as WAN Gateways. Updates from legacy replication servers are forwarded to all replication servers and not funneled to a single WAN Gateway. Also, updates from remote WAN Gateways will be forwarded to the legacy replication servers. Both of these actions will effectively increase WAN traffic during replication.

Recovering a Replication Changelog

In the event that the replication changelog is compromised (`<server-root>/changelogDb`), possibly due to a disk or NAS failure, perform the following steps.

1. Stop the server.
2. Backup `replicationChanges` from a remote server with the following command:

```
$ bin/backup --backupDirectory /app/backups/replicationChanges \
  --backendID replicationChanges
```

3. Copy the `replicationChanges` backup from the remote server and restore it on the local host as follows:

```
$ bin/restore --backupDirectory /app/tmp/replicationChanges
```

4. Start the server.

Disaster Recovery

In the event that data is compromised across all systems and a restore is necessary, perform the following steps. These steps assume that no read or write operations are performed by any servers during this process.

1. Stop all servers.
2. Run the following command on all servers:

```
$ /bin/dsreplication cleanup-local-server
```

3. Locate the backup or exported LDIF file that represents the last working copy of the database.
4. Restore the backup or import the LDIF file on a single server. If importing an LDIF file, use the `--excludeReplication` option with the `import-ldif` command.
5. Start the restored server. The server can now receive client requests.
6. Start another server in lockdown mode with the following command:

```
$ start-server --skipPrime --lockdownMode
```

7. Enable replication from the first server to the second server.
8. Initialize the second server from the first with the following command:

```
$ bin/dsreplication initialize
```

9. Restart the second server or use the `bin/leave-lockdown-mode` command to leave the server in lockdown mode. The second server can now receive client requests.
10. Repeat steps 6 through 9 for any other servers.

Chapter 20

Managing Logging

Topics:

The PingDirectory Server supports a rich set of log publishers to monitor access, debug, and error messages that occur during normal server processing. Administrators can view the standard set of default log files as well as configure custom log publishers with pre-defined filtering with its own log rotation and retention policies.

This chapter presents the following information:

Default Directory Server Logs

The Directory Server provides a standard set of default log files to monitor the server activity. You can view this set of logs in the `PingDirectory/logs` directory. The following default log files are available on the Directory Server and are presented below.

Table 59: Directory Server Logs

Log File	Description
access	File-based Access Log that records operations processed by the Directory Server. Access log records can be used to provide information about problems during operation processing (for example, unindexed searches, failed requests, etc.), and provide information about the time required to process each operation.
change-notifications.log	Records changes to data anywhere in the server, which match one or more configured change subscriptions.
config-audit.log	Records information about changes made to the Directory Server configuration in a format that can be replayed using the `dsconfig` tool
errors	File-based Error Log. Provides information about warnings, errors, and significant events that occur during server processing.
expensive-ops	Expensive Operations Log. Disabled by default. Provides only those operations that took longer than 1000 milliseconds to complete.
failed-ops	Failed Operations Log. Provides information on all operations that failed in single-line log format.
replication	Records any replication errors and publishes them to the filesystem.
searches-returning-no-entries	Searches Returning No Entries Log. Disabled by default. Provides information only on operations that did not return any entries.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to server.out.
server.pid	Stores the server’s process ID.
server.status	Stores the timestamp, a status code, and an optional message providing additional information on the server status.
setup.log	Records messages that occur during the initial configuration of a Directory Server with the `setup` tool.
tools	Directory that holds logs for long running utilities: `import-ldif`, `export-ldif`, `backup`, `restore`, `verify-index`. Current and previous copies of the log are present in the Directory Server.
update.log	Records messages that occur during a Directory Server update.

Types of Log Publishers

The PingDirectory Server provides several classes of log publishers for parsing, aggregating, and filtering information events that occur during normal processing in the server. There are three primary types of Log Publishers: access logs, debug logs, and error logs. Each type has multiple subtypes of log files based on the log server type:

Table 60: Types of Log Publishers

Log Publisher Type	Description	
Access	Provides the requests received and responses returned by the Directory Server. The information can be used to understand the operations performed by clients and to debug problems with the client applications. It can also be used for collecting usage information for performance and capacity planning purposes. There are tools described later that can analyze the access log to provide summaries of the LDAP activity and performance.	
	File-based Audit Log	Special type of access logger that provides detailed information about changes processed within the server. Disabled by default.
	JDBC-based Access Log	Stores access log information using a JDBC database connection. Disabled by default.
	File-based Access Logs	Provides a character-based stream used by TextWriter Publishers as a target for outputting log records. There are six types of file-based loggers: • **Admin Alert Access Log**. Generates administrative alerts for any operations that match the criteria for this access logger. Disabled by default. • **File-based Access Log**. Publishes access log messages to the filesystem. Enabled by default. • **Syslog-based Access Log**. Publishes access log messages to a syslogd port. Disabled by default. • **Expensive-Operations Access Log**. Publishes only those access log messages of operations that take longer than 1000 milliseconds. Disabled by default. • **Failed-Operations Access Log**. Publishes only those access log messages of operations that failed for any reason. Enabled by default. • **Successful Searches with No Entries Returned Log**. Publishes only those access log messages of search operations that failed to return any entries. Disabled by default.
Debug	Provides information about warnings, errors, or significant events that occur within the server.	
	Debug ACI Logger	Stores debug information on ACI evaluation for any request operations against the server
	File-based Error Logs	There are two types of File-based Error Logs: • **Error log**. Publishes error messages to the filesystem. Enabled by default. • **Replication log**. Publishes replication error messages to the filesystem. Enabled by default.
	JDBC-based Error Logs	Stores error log information using a JDBC database connection. Disabled by default.
	Syslog-based Error Logs	Publishes error messages to a syslogd port.

Viewing the List of Log Publishers

You can quickly view the list of log publishers on the Directory Server using the `dsconfig` tool.

Note: Initially, the JDBC, syslog, and Admin Alert log publishers must specifically be configured using `dsconfig` before they appear in the list of log publishers. Procedures to configure these types of log publishers appear later in this chapter.

To View the List of Log Publishers

- Use `dsconfig` to view the log publishers.

```
$ bin/dsconfig list-log-publishers
```

```
Log Publisher                                : Type              : enabled
---------------------------------------------:-------------------:--------
Debug ACI Logger                             : debug-access      : false
Expensive Operations Access Logger           : file-based-access : false
Failed Operations Access Logger              : file-based-access : true
File-Based Access Logger                     : file-based-access : true
File-Based Audit Logger                      : file-based-audit  : false
File-Based Debug Logger                      : file-based-debug  : false
File-Based Error Logger                      : file-based-error  : true
Replication Repair Logger                    : file-based-error  : true
Successful Searches with No Entries Returned : file-based-access : false
```

Enabling or Disabling a Default Log Publisher

You can enable or disable any log publisher available on the Directory Server using the `dsconfig` tool. By default, the following loggers are disabled and should be enabled only when troubleshooting an issue on the server:

Expensive Operations Access Logger
File-Based Audit Logger
File-Based Debug Logger
Successful Searches with No Entries Returned

To Enable a Default Access Log

- Use `dsconfig` to enable an Access Log Publisher. In this example, enable the Expensive-Ops log, which will record only those access log messages that take 1000 milliseconds or longer.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Expensive Operations Access Logger" --set enabled:true
```

Managing Access and Error Log Publishers

The Access Log records every request received and response returned by the Directory Server. The Access Log stores the IDs for the client connection, operation, the LDAP message involved with each client request, and the server response. The information can be used to debug any problems with a client application by correlating the numeric operation identifier to the client request or response.

The Directory Server supports multiple classes of access log publishers depending on your logging requirements. The following types of access log publishers are available on the system:

- **File-Based Access Log Publishers**. Provides a character-based TextWriter stream for outputting log records. There are three subclasses of TextWriter access logs:
  - **File-Based Access Logs**. Enabled by default. The File-based Access Log publishes access messages to the filesystem as `<server-root>/logs/access`. The Failed-Operations Log, Expensive-Operations Log, and the Searches with No Entries Returned Log are specialized types of the File-Based Access Log and shows only specific information necessary for troubleshooting purposes.

  - **Admin-Alert Access Logs**. Disabled by default. The Admin-Alert Access Log is specialized type of logger that automatically generates administrative alerts for any operations that match a criteria for this access log publisher.
  - **Syslog-Based Access Logs**. Disabled by default. The Syslog Access Log publishes access messages to a syslogd port.
- **File-Based Audit Logs**. Disabled by default. The Audit Log provides detailed information about modifications (writes) processed within the Directory Server. The File-based Audit Log publishes access messages to the filesystem as `<server-root>/logs/audit`.
- **JDBC-Based Access Logs**. Disabled by default. The JDBC-based Access Log provides information using a JDBC database connection.
- **JDBC-Based Error Logs**. Disabled by default. The JDBC-based Error Log provides information using a JDBC database connection.

Managing File-Based Access Log Publishers

Content-Reference to:

The PingDirectory Server supports a flexible and configurable Access Logging system that provides a full set of customized components to meet your system's logging requirements. The default Access Log can be configured to write information to a log file with two records per operation, one for the request received and one for the response returned. It can also be configured to write one message per operation or configured to record information about a subset of operations processed by the server. In addition to modifying existing default log files, you can create custom log publishers to monitor specific properties or connection criteria. For more information, see Creating New Log Publishers.

The Directory Server can be configured to use multiple access log publishers writing logs to different files using different configurations. This approach makes it possible to have fine-grained logging for various purposes (for example, a log that contains only failed operations, a log that contains only operations requested by root users, or a log that contains only operations that took longer than 20ms to complete).

The Directory Server provides an additional mechanism to filter access logs to record only a subset of messages of one or more types. The access log filtering mechanism uses the operation criteria (connection, request, result, search-entry, search-reference) to determine whether a given message should be logged based on information associated with the client connection as well as information in the operation request/response messages. For more information, see Configuring Filtered Logging.

Access Log Format

The Access Log has a standard format that lists various elements identifying the connection and operation occurring within the Directory Server. By default, each operation generates one access log message.

The Access Log displays the following common properties:

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Connection Type**. Displays the connection type requested by the client and the response by the server. Examples include the following:

  CONNECT
  BIND REQUEST/RESULT
  UNBIND REQUEST
  DISCONNECT
  SEARCH REQUEST/RESULT
  MODIFY REQUEST/RESPONSE
  others include: ABANDON, ADD, COMPARE, DELETE, EXTENDED OPERATION, MODIFY, MODIFY DN

- **Connection ID**. Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation. The connection ID is unique for a span of time on a single server. Values of the connection ID will be re-used when the server restarts or when it has had enough connections to cause the identifier to wrap back to zero.
- **Operation ID**. Numeric identifier, starting incrementally with 0, that identifies the operation. The operation ID is unique for a span of time on a single server. Values of the operation ID will be re-used when the server restarts or when it has serviced enough operations to cause the identifier to wrap back to zero.
- **Result Code**. LDAP result code that determines the success or failure of the operation result. Result messages include a result element that indicates whether the operation was successful or if failed, the general category for the failure, and an etime element that indicates the length of time in milliseconds that the server spent processing the operation.

  The Directory Server provides a useful tool `<server-root>/bin/ldap-result-code` (UNIX, Linux) or `<server-root>\bat\ldap-result-code` (Windows), that displays all of the result codes used in the system. You can use the utility if you are not sure what a result code means. For example, use the following:

  - `ldap-result-code` --list displays all of the defined result codes in the Directory Server.
  - `ldap-result-code` --int-value 16654 displays the name of the result code with a numeric value of 16654.
  - `ldap-result-code` --search operation displays a list of all result codes whose name includes the substring "operation".
- **Elapsed Time**. Displays the elapsed time (milliseconds) during which the operation completed its processing.
- **Message ID**. Numeric identifier, starting incrementally with 1, which identifies the LDAP message used to request the operation.

Access Log Example

The following example shows output from the Access Log in `<server-root>/logs/access`:

```
[01/Jun/2011:14:48:17 -0500] CONNECT conn=0 from="10.8.1.243" to="10.8.1.243"
 protocol="LDAP"
[01/Jun/2011:14:48:17 -0500] BIND REQUEST conn=0 op=0 msgID=1 version="3"
 dn="cn=Directory Manager"
  authType="SIMPLE"
[01/Jun/2011:14:48:17 -0500] BIND RESULT conn=0 op=0 msgID=1 resultCode=0
 etime=26.357
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
[01/Jun/2011:14:48:17 -0500] UNBIND REQUEST conn=0 op=1 msgID=2
[01/Jun/2011:14:48:17 -0500] DISCONNECT conn=0 reason="Client Unbind"
... (more output) ...
```

Modifying the Access Log Using dsconfig Interactive Mode

The File-Based Access Log can be modified to include or exclude all log messages of a given type using the `dsconfig` tool in interactive or non-interactive mode.

To Modify the Access Log Using dsconfig Interactive Mode

1. Use `dsconfig` in interactive mode to modify the access log properties.

   ```
   $ bin/dsconfig
   ```
2. Follow the prompts to specify the LDAP connection parameters for host name or IP address, connection type (LDAP, SSL, or StartTLS), port number, bind DN and password.
3. On the Directory Server main menu, type the number corresponding to the Log Publisher.
4. On the **Log Publisher Management** menu, enter the option to view and edit an existing log publisher.
5. On the **Log Publisher** menu, type the number corresponding to File-based Access Logger.
6. On the **File-Based Access Log Publisher** menu, type the number corresponding to the property that you want to change, and then follow the prompts.

7. Type `f` to apply the changes.

Modifying the Access Log Using dsconfig Non-Interactive Mode

You can use the `dsconfig` tool in non-interactive mode to quickly modify a log publisher property on the command line or in a script. For information on each property, see the *Directory Server Configuration Reference Guide*, which is an HTML document listing the various properties for each Directory Server component.

To Modify the Access Log Using dsconfig Non-Interactive Mode

- Use `dsconfig` with the --no-prompt option with the properties that you want to modify or set for your access log. In this example, enable the properties to include the instance name and startup ID.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Access
 Logger" \
  --set include-instance-name:true --set include-startup-id:true
```

Modifying the Maximum Length of Log Message Strings

By default, the Directory Server sets the maximum length of log message strings to 2000 characters. This value is configurable for any access log publisher, except the Syslog publisher, which is set to 500 characters. You can change the maximum length of log message strings by setting the `max-string-length` configuration property. If any string has more than the configured number of characters, then that string will be truncated and a placeholder will be appended to indicate the number of remaining characters in the original string.

To Modify the Maximum Length of Log Message Strings

- Use `dsconfig` to set the `max-string-length` property for an access log. The following command configures the "File-based Access Logger" to include the instance name and the maximum length of the log message strings to 5000 characters.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set include-instance-name:true \
  --set max-string-length:5000
```

Generating Access Logs Summaries

The Directory Server provides a convenience tool, `summarize-access-log`, that generates a summary of one or more file-based access logs. The summary provides analytical metric information that could be useful for administrators. The following metrics are provided in each summary:

- Total time span covered by the log files.
- Number of connections established and the average rate of new connections per second.
- IP addresses of up to the top 20 of the clients that most frequently connect to the server, the number of connections by each address, and the percentage of connections of each.
- Total number of operations processed, the number of operations of each type, the percentage of each type out of the total number, and the average rate per second for each type of operation.
- Average processing time for all operations and for each type of operation.
- Histogram of the processing times for each type of operation.
- Up to the 20 most common result codes for each type of operation, the number of operations of that type with that result code, and the percentage of operations of that type with that result code.
- Number of unindexed searches processed by the server.
- Breakdown of the scopes used for search operations with the number of percentage of searches with each scope.
- Breakdown of the most common entry counts for search operations with the number and percentage of searches that returned that number of entries.

- Breakdown of the most commonly used filter types for searches with a scope other than "base" (that is, those searches for which the server will use es when processing the filter). These filters will be represented in a generic manner so that any distinct assertion values or substring assertion elements will be replaced with question marks and attribute names in all lowercase characters (for example, `(givenName=John)` would be represented as `(givenName=?)`).

To Generate an Access Log Summary

- Use the `bin/summarize-access-log` with path to one or more access log files.

```
$ bin/summarize-access-log /path/to/logs/access
```

```
Examining access log /path/to/logs/access Examined 500 lines in 1 file
 covering a total duration of 1 day, 22 hours, 57 minutes, 31 seconds

Total connections established: 69 (0.000/second)
Total disconnects: 69 (0.000/second)

Most common client addresses:
127.0.0.1: 61 (88.406)
10.8.1.209: 8 (11.594)

Total operations examined: 181 ...
(metric for each operation examined) ...

Average operation processing duration: 22.727ms
Average add operation processing duration: 226.600ms
Average bind operation processing duration: 5.721ms
Average delete operation processing duration: 77.692ms
Average modify operation processing duration: 35.530ms
Average search operation processing duration: 4.017ms

Count of add operations by processing time:
... (histogram for add operations) ...

Count of bind operations by processing time:
... (histogram for bind operations) ...

Count of delete operations by processing time:
... (histogram for delete operations) ...

Count of modify operations by processing time:
... (histogram for modify operations) ...

Count of search operations by processing time:
... (histogram for search operations) ...

Most common add operation result codes:
success: 11 (84.615%)
entry already exists: 2 (15.385%)

Most common bind operation result codes:
success: 4 (50.000%)
invalid credentials: 4 (50.000%)

Most common delete operation result codes:
success: 1 (100.000%)

Most common modify operation result codes:
success: 9 (69.231%)
no such object: 4 (30.769%)
```

```
Most common search operation result codes:
success: 133 (91.724%)
no such object: 12 (8.276%)

Number of unindexed searches: 0

Most common search scopes:
BASE: 114 (78.621%)
SUB: 16 (11.034%)
ONE: 15 (10.345%)

Most common search entry counts:
1: 119 (82.069%)
0: 17 (11.724%)
2: 5 (3.448%)
10: 4 (2.759%)

Most common generic filters for searches with a non-base scope:
(objectclass=?): 19 (61.290%)
(ds-backend-id=?): 12 (38.710%)
```

About Log Compression

The Directory Server supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The `summarize-access-log` tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting purposes, administrators using compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the `compression-mechanism` property to have the value of "gzip" when creating a new logger.

About Log Signing

The Directory Server supports the ability to cryptographically sign a log to ensure that it has not been modified in any way. For example, financial institutions require audit logs for all transactions to check for correctness. Tamper-proof files are therefore needed to ensure that these transactions can be propertly validated and ensure that they have not been modified by any third-party entity or internally by unscrupulous employees. You can use the `dsconfig` tool to enable the `sign-log` property on a Log Publisher to turn on cryptographic signing.

When enabling signing for a logger that already exists and was enabled without signing, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the `validate-file-signature` tool provided in the `bin` directory of the server (or the `bat` directory for Windows systems).

Once you have enabled this property, you must disable and then re-enable the Log Publisher for the changes to take effect.

About Encrypting Log Files

The Directory Server supports the ability to encrypt log files as they are written. The `encrypt-log` configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an `.encrypted` extension (and if both encryption and compression are enabled, the extension will be `.gz.encrypted`). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The `encrypt-file` tool (or custom code, using the LDAP SDK's `com.unboundid.util.PassphraseEncryptedInputStream`) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the `encryption-settings create` command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the `encryption-settings-definition-id` property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the `rotate-log` tool to force the file to be rotated before attempting to examine it.

To Configure Log Signing

1. Use `dsconfig` to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set enabled:false
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set enabled:true
```

To Validate a Signed File

The Directory Server provides a tool, `validate-file-signature`, that checks if a file has not been tampered with in any way.

- Run the `validate-file-signature` tool to check if a signed file has been tampered with. For this example, assume that the `sign-log` property was enabled for the File-Based Audit Log Publisher.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

Note: If any validations errors occur, you will see a message similar to the one as follows:

```
One or more signature validation errors were encountered
while validating the contents of file 'logs/audit':
* The end of the input stream was encountered without
  encountering the end of an active signature block.
  The contents of this signed block cannot be trusted
  because the signature cannot be verified
```

To Configure Log File Encryption

1. Use `dsconfig` to enable encryption for a Log Publisher. In this example, the File-based Access Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted Access"
 \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-definition-
id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...Done
Writing decrypted data to file '/ds/PingDirectory/decrypted-access' using a
key generated from encryption settings definition
 '332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'
Successfully wrote 123,456,789 bytes of decrypted data
```

Creating New Log Publishers

The PingDirectory Server provides customization options to help you create your own log publishers with the `dsconfig` command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see Configuring Log Rotation and Configuring Log Retention.

To Create a New Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

Note: To configure compression on the logger, add the option to the previous command:

```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. If needed, view log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. On the main menu, select the option to configure the log publisher.
4. On the **Log Publisher** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the Directory Server root. For example, `logs/disconnect.log`.
9. Select the rotation policy to use for this log publisher.
10. Select the retention policy to use for this log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Type `f` to apply the changes.

Configuring Log Rotation

The Directory Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

To Configure the Log Rotation Policy

- Use `dsconfig` to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring Log Rotation Listeners

The Directory Server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed. Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true
```

The path specified by the `copy-to-directory` property must exist, and the filesystem containing that directory must have enough space to hold all of the log files that will be written there. The server will automatically monitor free disk space on the target filesystem and will generate administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the `summarize-access-log` tool on a recently-rotated log file and writes its output to a file in a specified location. This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the `summarize-access-log` tool, including the `file-based-access` and `operation-timing-access` logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of `.summary`. If the `output-directory` property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically be deleted. Though files are generally small in comparison to the log files themselves, make sure that there is enough space available in the specified storage directory. The server automatically monitors free disk space on the filesystem to which the summary files are written.

Configuring Log Retention

The Directory Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Directory Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Custom Retention Policy**. Create a new retention policy that meets your Directory Server’s requirements. This will require developing custom code to implement the desired log retention policy.

- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

- Use `dsconfig` to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configuring Filtered Logging

The PingDirectory Server provides a mechanism to filter access log messages based on specific criteria. The filtered log can then be used with a custom log publisher to create and to generate your own custom logs. Adding new filtered logs and associate publishers does not change the behavior of any existing logs. For example, adding a new log that only contains operations that were unsuccessful does not result in those operations being removed from the default access log.

The following example shows how to create a set of criteria that matches any operation that did not complete successfully. It then explains how to create a custom access log publisher that logs only operations matching that criteria. Note that this log does not include messages for connects or disconnects, and only a single message is logged per operation. This message contains both the request and result details.

To run log filtering based on any operation result (for example, result code, processing time, and response controls), turn off request logging and set the `include-request-details-in-result-messages` property to TRUE. Since filtering based on the results of an operation cannot be done until the operation completes, the server has no idea whether to log the request. Therefore, it might log request messages but not log any result messages. Instead, if you can only log result messages and include request details in the result messages, then only messages for operations that match the result criteria are logged. All pertinent information about the corresponding requests are included.

To Configure a Filtered Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create a result criteria object set to failure-result-codes, a predefined set of result codes that indicate when an operation did not complete successfully.

```
$ bin/dsconfig create-result-criteria --type simple \
  --criteria-name "Failed Operations" --set result-code-criteria:failure-
result-codes
```

2. Use `dsconfig` to create the corresponding log publisher that uses the result criteria. The log rotation and retention policies are also set with this command.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access \
  --publisher-name "Filtered Failed Operations" \
  --set enabled:true \
  --set log-connects:false \
  --set log-disconnects:false \
  --set log-requests:false \
  --set "result-criteria:Failed Operations" \
  --set log-file:logs/failed-ops.log \
  --set include-request-details-in-result-messages:true \
  --set "rotation-policy:7 Days Time Limit Rotation Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

3. View the `failed-ops.log` in the `logs` directory. Verify that only information about failed operations is written to it.

Managing Admin Alert Access Logs

Admin Alert Access Logs are a specialized form of filtered log that automatically generates an administrative alert when criteria configured for the log publisher matches those messages in the access log.

About Access Log Criteria

Configuring an Admin Alert Access Log requires that you configure the criteria for the access log messages. Each criteria can be either a Simple or an Aggregate type. The Simple type uses the set of properties for the client connection, operation request, and the contents of any operation-specific requests or results. The Aggregate type provides criteria that contains Boolean combination of other operation-specific criteria objects. For more information, see the *Ping Identity Directory Server Configuration Reference*.

The criteria can be one or more of the following:

- **Connection Criteria**. Defines sets of criteria for grouping and describing client connections based on a number of properties, including protocol, client address, connection security, and authentication state.
- **Request Criteria**. Defines sets of criteria for grouping and describing operation requests based on a number of properties, including properties for the associated client connection, the type of operation, targeted entry, request controls, target attributes, and other operation-specific terms.
- **Result Criteria**. Defines sets of criteria for grouping and describing operation results based on a number of properties, including the associated client connection and operation request, the result code, response controls, privileges missing or used, and other operation-specific terms.
- **Search Entry Criteria**. Defines sets of criteria for grouping and describing search result entries based on a number of properties, including the associated client connection and operation request, the entry location and contents, and included controls.
- **Search Reference Criteria**. Defines sets of criteria for grouping and describing search result references based on a number of properties, including the associated client connection and operation request, reference contents, and included controls.

Configuring an Admin Alert Access Log Publisher

Prior to configuring an Admin Alert Access Log, you must establish an administrative alert handler in your system. For more information, see Working with Administrative Alert Handlers.

To Configure an Admin Alert Access Log Publisher

1. Use `dsconfig` to create a criteria object for the Admin Alert Access Log. For this example, we want to log only write operations that target user entries. The following command matches any of the specified operations whose target entry matches the filter “`(objectClass=person)`”.

   If you are using the `dsconfig` tool in interactive mode, the menu items for the criteria operations are located in the Standard objects menu.

```
$ bin/dsconfig create-request-criteria --type simple \
  --criteria-name “User Updates” \
  --set operation-type:add \
  --set operation-type:delete \
  --set operation-type:modify \
  --set operation-type:modify-dn \
  --set “any-included-target-entry-filter:(objectClass=person)”
```

2. Use `dsconfig` to create a log publisher of type `admin-alert-access`.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name ”User Updates Admin Alert Access Log” \
  --type admin-alert-access \
  --set “request-criteria:User Updates” \
  --set include-request-details-in-result-messages:true \
  --set enabled:true
```

Managing Syslog-Based Access Log Publishers

The Directory Server supports access logging using the syslog protocol that has been part of the Berkeley Software Distribution (BSD) operating systems for many years. Syslog provides a flexible, albeit simple, means to generate, store and transfer log messages that is supported on most UNIX and Linux operating systems.

The quasi-standard syslog message format cannot exceed 1 kbytes and has three important parts:

- **PRI**. Specifies the message priority based on its facility and severity. The message facility is a numeric identifier that specifies the type of log messages, such as kernel messages, mail system messages, etc. The severity is a numeric identifier that specifies the severity level of the operation that is being reported. Together, the facility and the severity determine the priority of the log message indicated by angled brackets and 1-3 digit priority number. For example, "<0>", "<13>", "<103>" are valid representations of the PRI.
- **Timestamp and Host Name**. The timestamp displays the current date and time of the log. The host name or IP address displays the source of the log.
- **Message**. Displays the actual log message.

Administrators can configure syslog to handle log messages using log priorities that are based on the message's facility and severity. This feature allows users to configure the logging system in such a way that messages with high severities can be sent to a centralized repository, while lower severity messages can be stored locally on a server.

Note: Since the numeric values of the severity and facility are operating system-dependent, the central repository must only include syslog messages from compatible OS types, otherwise the meanings of the PRI field is ambiguous.

Before You Begin

You will need to identify the host name and port to which you want to connect. Because the Syslog Protocol uses User Datagram Protocol (UDP) packets, we highly recommend that you use `localhost` and utilize some additional logging tools, such as `syslog-ng`. UDP is unreliable and unsecure means to transfer data packets between hosts.

Default Access Log Severity Level

All messages are logged at the syslog severity level of 6, which is `Informational: informational` messages. Note that this value is not standard across different types of UNIX or Linux systems. Please consult your particular operating system.

Syslog Facility Properties

When using syslog, specify a facility for the access log messages. As an advanced property, you can select a number that corresponds to the facility you wish to use. The default value for the `syslog-facility` property is 1 (one) for user-level messages. Note that these values are not standard across different types of UNIX or Linux systems. Please consult your particular operating system documentation for properties specific to that system.

Table 61: Syslog Facility Properties

Facility	Description
0	kernel messages
1	user-level messages (default)
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem

Facility	Description
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local use 0
17	local use 1
18	local use 2
19	local use 3
20	local use 4
21	local use 5
22	local use 6
23	local use 7

Queue-Size Property

The maximum number of log records that can be stored in the asynchronous queue is determined by the `queue-size` property. The default queue size set is 10000, which means that the server will continuously flush messages from the queue to the log. The server does not wait for the queue to fill up before flushing to the log. Therefore, lowering this value can impact performance.

Configuring a Syslog-Based Access Log Publisher

You can configure a Syslog-based Access Log Publisher using the `dsconfig` tool. We recommend that you use syslog locally on localhost and use `syslog-ng` to send the syslog messages to remote syslog servers.

Because syslog implementations differ by vendor, please review your particular vendor's syslog configuration.

To Configure a Syslog-Based Access Log Publisher

Content-Reference to:

- Use `dsconfig` to create a log publisher of type syslog-based-access.

  If you are using the `dsconfig` tool in interactive mode, the menu item for Syslog Facility is an advanced property, which can be exposed by typing `a` (for "show advanced properties") on the Syslog-Based Access Log Publisher menu.

  ```
  $ bin/dsconfig create-log-publisher \
  --publisher-name "syslog-access" \
  --type syslog-based-access \
  --set syslog-facility:4 \
  --set enabled:true
  ```

Managing the File-Based Audit Log Publishers

The Directory Server provides an audit log, a specialized version of the access log, for troubleshooting problems that may occur in the course of processing. The log records all changes to the data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

Audit Log Format

The audit log uses standard LDIF format, so that administrators can quickly analyze what changes occurred to the data. The audit log begins logging when enabled and should be used to debug any issues that may have occurred. Some common properties are the following:

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Connection ID**. Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation.
- **Operation ID**. Numeric identifier, starting incrementally with 0, that identifies the operation.
- **Modifiers Name**. Displays the DN of the user who made the change.
- **Update Time**. Records the `modifyTimestamp` operational attribute.

Audit Log Example

The following example shows output from the Audit Log in the `<server-root>/ logs/audit`. The first entry shows when the audit log was enabled. The second entry show changes made to a user entry.

```
# 05/Jun/2011:10:29:04 -0500; conn=0; op=55
dn: cn=File-Based Audit Logger,cn=Loggers,cn=config
changetype: modify
replace: ds-cfg-enabled
ds-cfg-enabled: true
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z

# 05/Jun/2011:10:31:20 -0500; conn=2; op=1
dn: uid=user.996,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 115 426 4748
-
replace: homePhone
homePhone: +1 407 383 4949
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
```

Enabling the File-Based Audit Log Publisher

You can enable the File-Based Audit Log Publisher using the `dsconfig` tool. The audit log can impact the Directory Server's write performance, so enable it only when troubleshooting any issues.

To Enable the File-Based Audit Log Publisher

- Use `dsconfig` to enable the File-Based Audit Log Publisher. For this example, the instance name and startup ID is also enabled in the audit log.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true \
  --set include-instance-name:true \
  --set include-startup-id:true
```

Obscuring Values in the Audit Log

You can obscure the values of specific attributes in the audit log using the `obscure-attribute` property. Each value of an obscured attribute is replaced in the audit log with a string of the form "***** OBSCURED VALUE *****". By default, attributes are not obscured, because the values of password attributes appear in hashed form rather than in the clear.

Managing the JDBC Access Log Publishers

Content-Reference to:

The Directory Server supports the Java Database Connectivity (JDBCTM) API, which allows access to SQL data stores by means of its JDBC drivers. The JDBC 4.0 API, part of the Java SDK, provides a seamless method to interface with various database types in heterogeneous environments.

By easily connecting to a database, the Directory Server can be configured to implement a centralized logging system with different databases. Centralized logging simplifies log correlation and analysis tasks and provides security by storing data in a single repository. Some disadvantages of centralized logging are that data flow asymmetries may complicate synchronization or network provisioning and may unduly burden the central repository with possibly heavy loads.

Before You Begin

Before configuring the JDBC Access Log Publisher, you need to carry out two essential steps to set up the database.

- Install the database drivers in the Directory Server `lib` directory.
- Define the log mapping tables needed to map access log elements to the database column data. Only those elements in the log mapping table gets logged by the JDBC Log Publisher.

The following sections provide more information about these tasks.

Configuring the JDBC Drivers

The Directory Server supports a number of JDBC drivers available in the market. We highly recommend using the JDBC 4 drivers supported in the Java platform. For example, for Oracle databases, you need to use the `ojdbc.jar` driver for Java and any associated JAR files (National Language Support jars, and others) required to connect with the particular database. The following databases are supported:

DB2
MySQL
Oracle Call Interface (OCI)
Oracle Thin
PostgreSQL

SQL Server

To Configure the JDBC Driver

- Obtain the JAR file(s) for your particular database, and copy it into the `<server-root>/lib` directory.

Configuring the Log Field Mapping Tables

The log field mapping table associates access log fields to the database column names. You can configure the log field mapping table using the `dsconfig` tool, which then generates a DDL file that you can import into your database. The DDL file is generated when you create the JDBC Log Publisher.

To uniquely identify a log record, we recommend always mapping the following fields: `timestamp`, `startupid`, `message-type`, `connection-id`, `operation-type`, `instance-name`.

The table name is not part of this mapping.

The Directory Server also provides three options that you can quickly select for creating a log field mapping table:

- **Complete JDBC Access Log Field Mappings**. Maps all 52 object properties.
- **Complete JDBC Error Log Field Mappings**. Maps all 8 object properties.
- **Simple JDBC Access Log Field Mappings**. Maps a common set of object properties.
- **Custom JDBC Access Log Field Mappings**. Create a custom set of JDBC log field mappings.
- **Custom JDBC Error Log Field Mappings**. Create a custom set of JDBC error log field mappings.

To Configure the Log Field Mapping Tables

Content-Reference to:

1. Use `dsconfig` to create a log field mapping table. On the main menu, type `o` to change to the Standard Object menu, and type the number corresponding to Log Field Mapping.
2. On the **Log Field Mapping management** menu, enter the option to create a new Log Field Mapping.
3. On the **Log Field Mapping template** menu, enter the option to select a complete JDBC Access Log Field mapping to use as a template for your new field mapping.
4. Next, enter a name for the new field mapping. In this example, type `my-jdbc-test`.
5. On the **Access Log Field Mapping Properties** menu, select a property for which you want to change the value. Any property that is undefined will not be logged by the JDBC Access Log Publisher. When complete, type `f` to save and apply the changes.
6. On the **Log Field Mapping Management** menu, type `q` to exit the menu.
7. View the existing Log Mappings on the system.

```
$ bin/dsconfig list-log-field-mappings
```

```
Log Field Mapping                    : Type
-------------------------------------:-------
Complete JDBC Access Log Field Mappings : access
Complete JDBC Error Log Field Mappings  : error
my-jdbc-test                            : access
Simple JDBC Access Log Field Mappings   : access
```

Configuring the JDBC Access Log Publisher using dsconfig Interactive Mode

After setting up the drivers and the log mapping table, use the `dsconfig` utility to configure the JDBC Access Log Publisher on the Directory Server. The following example uses `dsconfig` interactive mode to illustrate the steps required to configure the log publisher and the external database server.

To Configure the JDBC Access Log Publisher

1. Copy the database JAR files to the `<server-root>/lib` directory, and then restart the Directory Server.

2. Launch the `dsconfig` tool in interactive command-line mode.

```
$ bin/dsconfig
```

3. Next, type the connection parameters to bind to the Directory Server. Enter the host name or IP address, type of LDAP connection (LDAP, SSL, or StartTLS) that you are using on the Directory Server, the LDAP listener port number, the user bind DN, and the bind DN password.
4. On the main menu, type the number corresponding to Log Publisher.
5. On the **Log Publisher management** menu, enter the option to create a new Log Publisher.
6. On the **Log Publisher template** menu, type n to create a new Log Publisher.
7. On the **Log Publisher Type** menu, enter the option to create a new JDBC-Based Access Log Publisher.
8. Type a name for the JDBC Access Log Publisher.
9. On the **Enabled Property** menu, enter the option to enable the log publisher.
10. On the **Server Property** menu, enter the option to create a new JDBC External Server.
11. Next, type the name for the JDBC External Server. This is a symbolic name used to represent the DBMS.
12. On the **JDBC Driver Type Property** menu, type the number corresponding to the type of JDBC database driver type.
13. Next, type a name for the `database-name` property. This is the DBMS database name. The database name must contain the table referred to in the generated DDL.
14. Next, type the host name or IP address (server-host-name) of the external server.
15. Type the server listener port. In this example, type `1541`.
16. Review the properties for the external server, and the type `f` to apply the changes.
17. If you need to supply your own JDBC URL, type a for advanced properties to open the `jdbc-driver-url` property and supply the appropriate URL. The example below shows how to access an Oracle Thin Client connection using a SID instead of a Service.

```
>>>> Configure the properties of the JDBC External Server

Property Value(s)
-----------------------------------------------------
1) description -
2) jdbc-driver-type oraclethin
3) jdbc-driver-url jdbc:oracle:thin@myhost:1541:my_SID
4) database-name jdbc-test
5) server-host-name localhost
6) server-port 1541
7) user-name -
8) password -

?) help
f) finish - create the new JDBC External Server
a) hide advanced properties of the JDBC External Server
d) display the equivalent dsconfig arguments to create this object
b) back
q) quit

Enter choice [b]: f
```

```
JDBC External Server was created successfully
```

18. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `<server-root>/logs/ddls/<name-of-logger>.sql` file. Import the DDL file to your database.

Configuring the JDBC Access Log Publisher Using dsconfig Non-Interactive Mode

The following example uses `dsconfig` non-interactive mode to illustrate the steps to configure the log publisher and the external database server presented in the previous section.

To Configure the JDBC Access Log Publisher in Non-Interactive Mode

1. Use `dsconfig` with the --no-prompt option to create the JDBC external server.

   ```
   $ bin/dsconfig --no-prompt create-external-server \
     --server-name jdbc-external \ --type jdbc \
     --set jdbc-driver-type:oraclethin \
     --set database-name:ubid_access_log \
     --set server-host-name:localhost --set server-port:1541
   ```

2. Use `dsconfig` to create the log publisher.

   ```
   $ bin/dsconfig --no-prompt create-log-publisher \
     --publisher-name jdbc-test \
     --type jdbc-based-access \
     --set enabled:true \
     --set server:jdbc-external \
     --set "log-field-mapping:Simple JDBC Access Log Field Mappings"
   ```

3. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `&lt;server-root>/logs/ddls/&lt;name- of-logger>.sql` file. Import the DDL file to your database.

   The procedure to configure the JDBC-Based Error Log Publisher is similar to creating a JDBC-Based Access Log Publisher. You can run the previous `dsconfig` command with the --type jdbc-based-error as follows:

   ```
   $ bin/dsconfig --no-prompt create-log-publisher \
   --publisher-name jdbc-error-test \
   --type jdbc-based-error \
   --set enabled:true \
   --set server:jdbc-external \
   --set "log-field-mapping:Simple JDBC Access Log Field Mappings"
   ```

Managing the File-Based Error Log Publisher

The Error Log reports errors, warnings, and informational messages about events that occur during the course of the Directory Server's operation. Each entry in the error log records the following properties (some are disabled by default and must be enabled):

- **Time Stamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Category**. Specifies the message category that is loosely based on the server components.
- **Severity**. Specifies the message severity of the event, which defines the importance of the message in terms of major errors that need to be quickly addressed. The default severity levels are: `fatal-error`, `notice`, `severe-error`, `severe-warning`.
- **Message ID**. Specifies the numeric identifier of the message.
- **Message**. Stores the error, warning, or informational message.

Error Log Example

The following example displays the error log for the Data Metrics Server. The log is enabled by default and is accessible in the `<server-root>/logs/errors` file.

```
[21/Oct/2012:05:15:23.048 -0500] category=RUNTIME_INFORMATION severity=NOTICE
msgID=20381715 msg="JVM Arguments: '-Xmx8g', '-Xms8g', '-XX:MaxNewSize=1g',
'-XX:NewSize=1g', '-XX:+UseConcMarkSweepGC', '-XX:+CMSConcurrentMTEnabled',
```

```
'-XX:+CMSParallelRemarkEnabled', '-XX:+CMSParallelSurvivorRemarkEnabled',
'-XX:+CMSScavengeBeforeRemark', '-XX:RefDiscoveryPolicy=1',
'-XX:ParallelCMSThreads=4', '-XX:CMSMaxAbortablePrecleanTime=3600000',
'-XX:CMSInitiatingOccupancyFraction=80', '-XX:+UseParNewGC', '-XX:
+UseMembar',
'-XX:+UseBiasedLocking', '-XX:+UseLargePages', '-XX:+UseCompressedOops',
'-XX:PermSize=128M', '-XX:+HeapDumpOnOutOfMemoryError',
'-Dcom.unboundid.directory.server.scriptName=setup'"
[21/Oct/2012:05:15:23.081 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555611 msg="Administrative alert type=server-starting
id=4178daee-ba3a-4be5-8e07-5ba17bf30b71
class=com.unboundid.directory.server.core.MetricsEngine
msg='The Data Metrics Server is starting'"
[21/Oct/2012:05:15:23.585 -0500] category=CORE severity=NOTICE
msgID=1879507338 msg="Starting group processing for backend api-users"
[21/Oct/2012:05:15:23.586 -0500] category=CORE severity=NOTICE
msgID=1879507339 msg="Completed group processing for backend api-users"
[21/Oct/2012:05:15:23.586 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555575 msg="'Group cache (2 static group(s) with 0 total
memberships and 0 unique members, 0 virtual static group(s),
1 dynamic group(s))' currently consumes 7968 bytes and can grow to a maximum
of an unknown number of bytes"
[21/Oct/2012:05:16:18.011 -0500] category=CORE severity=NOTICE
msgID=458887 msg="The Data Metrics Server (Data Metrics Server 7.0.0.0
build 20121021003738Z, R12799) has started successfully"
```

To Modify the File-Based Error Logs

- Use `dsconfig` to modify the default File-Based Error Log.

  ```
  $ bin/dsconfig set-log-publisher-prop \
    --publisher-name "File-Based Error Logger" \
    --set include-product-name:true --set include-instance-name:true \
    --set include-startup-id:true
  ```

Managing the Syslog-Based Error Log Publisher

The Directory Server supports a Syslog-Based Error Log Publisher using the same mechanism as the Syslog-Based Access Log Publisher. You can easily configure the error logger using the `dsconfig` tool.

Syslog Error Mapping

The Directory Server automatically maps error log severities to the syslog severities. The following mappings are used:

Table 62: Error to Syslog Severities Mappings to Syslog

Error Log Facility	Syslog Severity
FATAL_ERROR,0	Syslog Emergency
SEVERE_ERROR,1	Syslog Alert
SEVERE_WARNING,2	Syslog Critical
MILD_ERROR,3	Syslog Error
MILD_WARNING,4	Syslog Warn
NOTICE,5	Syslog Notice
INFORMATION,6	Syslog Info

Error Log Facility	Syslog Severity
DEBUG,7	Syslog Debug

Configuring a Syslog-Based Error Log Publisher

You can configure a Syslog-based Error Log Publisher using the `dsconfig` tool. Again, we recommend that you use syslog locally on `localhost` and use `syslog-ng` to send the data packets over the UDP protocol.

Because syslog implementations differ by vendor, please review your particular vendor's syslog configuration.

To Configure a Syslog-Based Error Log Publisher

- Use `dsconfig` to create a log publisher of type `syslog-based-error`. In this example, set the syslog facility to 4 for security/authorization messages.

```
$ bin/dsconfig create-log-publisher --publisher-name "syslog-error" \
  --type syslog-based-error --set syslog-facility:4 --set enabled:true
```

Creating File-Based Debug Log Publishers

The Directory Server provides a File-Based Debug Log Publisher that can be configured when troubleshooting a problem that might occur during server processing. Because the debug data may be too large to maintain during normal operations, the Debug Log Publisher must be specifically configured and enabled. The Debug Log reports the following types of information:

Exception data thrown by the server.
Data read or written to network clients.
Data read or written to the database.
Access control or password policy data made within the server.

You can use the `dsconfig` tool to create a debug log publisher. You should only create a debug logger when troubleshooting a problem due to the voluminous output the Directory Server generates.

To Create a File-Based Debug Log Publisher

- Use `dsconfig` to create the debug log publisher. The `log-file` property (required) sets the debug log path. You must also specify the rotation and retention policy for the debug log.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "File-Based Debug Logger" \
  --type file-based-debug \
  --set enabled:true \
  --set log-file:/logs/debug \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy"
```

To Delete a File-Based Debug Log Publisher

- Use `dsconfig` to create the debug log publisher.

```
$ bin/dsconfig delete-log-publisher \
  --publisher-name "File-Based Debug Logger"
```

Chapter

21

Managing Monitoring

Topics:

The PingDirectory Server also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the PingData® Data Metrics Server, the Administrative Console, SNMP, JMX, and over LDAP. The Directory Server also provides a tool, the Periodic Stats Logger, to profile server performance.

This chapter presents the following information:

- *Profiling Server Performance Using the Stats Logger*
- *Adding Custom Logged Statistics to a Periodic Stats Logger*

The Monitor Backend

The Directory Server exposes its monitoring information under the `cn=monitor` entry. Administrators can use various means to monitor the servers, including the PingData Data Metrics Server, through SNMP, the Administrative Console, JConsole, LDAP command-line tools, and the Periodic Stats Logger. Use the `bin/status` tool to display server component activity and state.

The list of all monitor entries can be seen using ldapsearch as follows:

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
 --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
 --baseDN "cn=monitor" "(objectclass=*)" cn
```

The following table describes a subset of the monitor entries:

Table 63: Directory Server Monitoring Components

Component	Description
Active Operations	Provides information about the operations currently being processed by the Directory Server. Shows the number of operations, information on each operation, and the number of active persistent searches.
Backends	Provides general information about the state of an a Directory Server backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size.
Client Connections	Provides information about all client connections to the Directory Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., connID="15", connectTime="20100308223038Z", etc.)
Connection Handlers	Provides information about the available connection handlers on the Directory Server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
Disk Space Usage	Provides information about the disk space available to various components of the Directory Server.
General	Provides general information about the state of the Directory Server, including product name, vendor name, server version, etc.
Index	Provides on each index. The monitor captures the number of keys preloaded, and counters for read/write/remove/open-cursor/read-for-search. These counters provide insight into how useful an index is for a given workload.
HTTP/HTTPS Connection Handler Statistics	Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc.

Component	Description
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc.
System Information	Provides general information about the system and the JVM on which the Directory Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Directory Server version, including build ID, version, revision number, etc.
Work Queue	Provides information about the state of the Directory Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, and number of busy worker threads. The work queue configuration has a `monitor-queue-time` property set to `true` by default. This logs messages for new operations with a `qtime` attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

Monitoring Disk Space Usage

The disk space usage monitor provides information about the amount of usable disk space available for Directory Server components. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface by various components of the server. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `/config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the DiskSpaceConsumer interface.

The disk space usage monitor provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

Three thresholds can be configured for this monitor:

- **Low space warning threshold**. This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Directory Server will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The Directory Server will also generate additional alerts as the amount of usable space is further reduced (e.g., each time the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.
- **Low space error threshold**. This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. Once the server enters this mode, then an administrator will have to take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server

will resume normal operation. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.

- **Out of space error threshold**. This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the PingDirectory Server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.
- **Disk space monitoring for tools**. The server monitors disk space consumption during processing for the `export-ldif`, `rebuild-index`, and `backup` tools. Space is monitored every 10 seconds if usable space for all monitored paths is greater than 15 percent of the capacity of those volumes. If usable space for any path drops below 15 percent, or below 10GB free, the space check frequency is increased to every second. Warning messages are generated if available space falls below 10 percent, or below 5GB free. If usable space for any path drops below two percent, or 1GB free, the tool processing is aborted and files may be removed to free up space.

The default configuration uses the same values for the low space error threshold and out of space error threshold. This is to prevent having the server online but rejecting requests, which will cause problems with applications trying to interact with the server. The low space warning threshold generates an alert before the problem becomes serious, well in advance of available disk space dropping to a point that it is critical.

The default values may not be suitable for all disk sizes, and should be adjusted to fit the deployment. Determining the best values should factor in the size of the disk, how big the database may become, how much space log files may consume, and how many backups will be stored.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring with the Data Metrics Server

The Data Metrics Server is an invaluable tool for collecting, aggregating and exposing historical and instantaneous data from the various Ping Identity servers in a deployment. The Data Metrics Server relies on a captive PostgreSQL data store for the metrics, which it collects from internal instrumentation across the instances, replicas, and data centers in your environment. The data is available via a Monitoring API that can be used to build custom dashboards and monitoring applications to monitor the overall health of your PingData Platform system. For more information, see the *PingData Data Metrics Server Administration Guide*.

About the Collection of System Monitoring Data

All PingData servers have the capability to monitor the health of the server and host system they run on for diagnostic review and troubleshooting. Initially, the servers do not collect any performance data until they are prepared for monitoring by a PingData Data Metrics Server using the `monitored-servers add-servers` tool or an administrator enables system health data collection for real-time inspection and querying. At a high level, all of the important server and machine metrics which can be monitored are available in the `cn=monitor` backend.

The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics. Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and what kind of data aggregation occurs for LDAP application statistics. The Stats Collector plugin ensures that a Data Metrics Server is able to gather all of the detailed data required for a comprehensive diagnostic review.

The Stats Collector plugin relies exclusively on entries in the `cn=monitor` backend to sample data using LDAP queries. The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics. Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and the type of data aggregation that occurs for LDAP application statistics. The Stats Collector plugin is configured with the `dsconfig` tool and collects data using LDAP queries. For example, the `--server-info:extended` option includes collection for the following:

- CPU
- JVM memory
- Memory
- Disk information
- Network information

Utilization metrics are gathered via externally invoked OS commands, such as `iostat` and `netstat`, using platform-specific arguments and version-specific output parsing.

Enabling the Host System monitor provider automatically gathers CPU and memory utilization but only optionally gathers disk and network information. Disk and network interfaces are enumerated in the configuration by device names (e.g., `eth0` or lo), and by disk device names (e.g., `sd1, sdab, sda2, scsi0`).

Monitoring Key Performance Indicators by Application

The PingDirectory Server can be configured to track many key performance metrics (for example, throughput and response-time) by the client applications requesting them. This feature is invaluable for measuring whether the Ping Identity identify infrastructure meets all of your service-level agreements (SLA) that have been defined for client applications.

When enabled, the per-application monitoring data can be accessed in the `cn=monitor` backend, the Periodic Stats Logger, and made available for collection by the PingData Data Metrics Server. See the “Profiling Server Performance Using the Periodic Stats Logger” for more information on using that component. Also, see the Directory Server Configuration section of the *PingData Data Metrics Server Administration Guide* for details on configuring the server to expose metrics that interest you. Tracked application information is exposed in the Data Metrics Server by metrics having the 'application-name' dimension. See the documentation under `docs/metrics` of the Data Metrics Server for information on which metrics are available with the 'application-name' dimension.

Configuring the External Servers

Before you install the Data Metrics Server, you need to configure the servers you will be monitoring: PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. The Data Metrics Server requires all servers to be version 3.5.0 or later. See the administration guides for each product for installation instructions.

Once you have installed the Directory Server, you can use the `dsconfig` tool to make configuration changes for the Data Metrics Server. When using the `dsconfig` tool interactively, set the complexity level to Advanced, so that you can make all the necessary configuration changes.

Preparing the Servers Monitored by the Data Metrics Server

The Metrics Backend manages the storage of metrics and provides access to the stored blocks of metrics via LDAP. The Metrics Backend is configured to keep a maximum amount of metric history based on log retention policies. The default retention policy uses the Default Size Limit Retention Policy, Free Disk Space Retention Policy, and the File Growth Limit Policy, limiting the total disk space used to 500 MB. This amount of disk typically contains more than 24 hours of metric history, which is ample. The Directory Server keeps a metric history so that the Data Metrics Server can be down for a period and then catch up when it comes back online.

The following two commands create a Retention Policy that limits the number of files to 2000, and sets the Metrics Backend to flush data to a new file every 30 seconds.

```
$ bin/dsconfig create-log-retention-policy \
  --policy-name StatsCollectorRetentionPolicy \
  --type file-count --set number-of-files:2000

$ bin/dsconfig set-backend-prop \
  --backend-name metrics --set sample-flush-interval:30s \
  --set retention-policy:StatsCollectorRetentionPolicy
```

These commands configure the Metrics Backend to keep 16 hours of metric history, which consumes about 250 MB of disk, ensuring that captured metrics are available to the Data Metrics Server within 30 seconds of when the metric was captured. The value of the `sample-flush-interval` attribute determines the maximum delay between when a metric is captured and when it can be picked up by the Data Metrics Server.

The flush interval can be set between 15 seconds and 60 seconds, with longer values resulting in less processing load on the Data Metrics Server. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in the Dashboard Application. If you change the `sample-flush-interval` attribute to 60 seconds in the example above, then the Directory Server keeps 2000 minutes of history. Because the number of metrics produced per unit of time can vary depending on the configuration, no exact formula can be used to compute how much storage is required for each hour of history. However, 20 MB per hour is a good estimate.

Configuring the Processing Time Histogram Plugin

The Processing Time Histogram plugin is configured on each Directory Server and Directory Proxy Server as a set of histogram bucket ranges. When the bucket ranges for a histogram change, the Data Metrics Server notices the change and marks samples differently. This process allows for histograms with the same set of bucket definitions to be properly aggregated and understood when returned in a query. If different servers have different bucket definitions, then a single metric query cannot return histogram data from the servers.

You should try to keep the Processing Time Histogram bucket definitions the same on all servers. Having different definitions restricts the ability of the Data Metrics Server API to aggregate histogram data across servers and makes the results of a query asking "What percentage of the search requests took less than 12 milliseconds?" harder to understand.

For each server in your topology, you must set the separate-monitor-entry-per-tracked-application property of the processing time histogram plugin to true. This property must be set to expose per-application monitoring information under `cn=monitor`. When the separate-monitor-entry-per-tracked-application property is set to true, then the per-application-ldap-stats property must be set to per-application-only on the Stats Collector Plugin and vice versa.

For example, the following `dsconfig` command line sets the required properties of the Processing Time Histogram plugin:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Processing Time Histogram” \
  --set separate-monitor-entry-per-tracked-application:true
```

The following `dsconfig` command line sets the per-application-ldap-stats property of the Stats Collector plugin to per-application-only:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Stats Collector” \
 --set per-application-ldap-stats:per-application-only
```

Setting the Connection Criteria to Collect SLA Statistics by Application

If you want to collect data about your SLAs, you need to configure connection criteria for each Service Level Agreement that you want to track. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used when the server needs to perform matching based on connection-level properties, such as filtered logging. For assistance using connection criteria, contact your authorized support provider.

For example, imagine that we are interested in collecting statistics on data that is accessed by clients authenticating as the Directory Manager. We need to create connection criteria on the Directory Server that identifies any user authenticating as the Directory Manager. The connection criteria name corresponds to the application-name dimension value that clients will specify when accessing the data via the API. When you define the Connection Criteria, change the included-user-base-dn property to include the Directory Manager’s full LDIF entry.

The following `dsconfig` command line creates connection criteria for the Directory Manager:

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name “Directory Manager” \
  --type simple \
  --set “included-user-base-dn:cn=Directory Manager,cn=Root DNs,cn=config”
```

Updating the Global Configuration

You also need to create Global Configuration-tracked applications for each app (connection criteria) you intend to track. The `tracked-application` property allows individual applications to be identified in the server by connection criteria. The name of the tracked application is the same as the name you defined for the connection criteria.

For example, the following `dsconfig` command line adds the connection criteria we created in the previous step to the list of tracked applications:

```
$ bin/dsconfig set-global-configuration-prop \
  --set "tracked-application:Directory Manager”
```

The value of the `tracked-application` field corresponds to the value of the `application-name` dimension value that clients will specify when accessing the data via the API.

Proxy Considerations for Tracked Applications

In a proxy environment, the criteria should be defined in the Directory Proxy Server since the Directory Proxy Server passes the application name through to the Directory Server in the intermediate client control. If a client of the Directory Proxy Server or Directory Server happens to use the intermediate client control, then the client name specified in the control will be used as the application name regardless of the criteria listed in the tracked-application property.

Monitoring Using SNMP

The PingDirectory Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems, such as Alcatel-Lucent (Omnivista EMS), HP (OpenView), IBM-Tivoli (Netview), Oracle-Sun (Solstice Enterprise Manager), and others. Specific discussion on integrating an SNMP deployment on an NMS system is beyond the scope of this chapter. Consult with your NMS system for specific information.

The PingDirectory Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Server’s SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

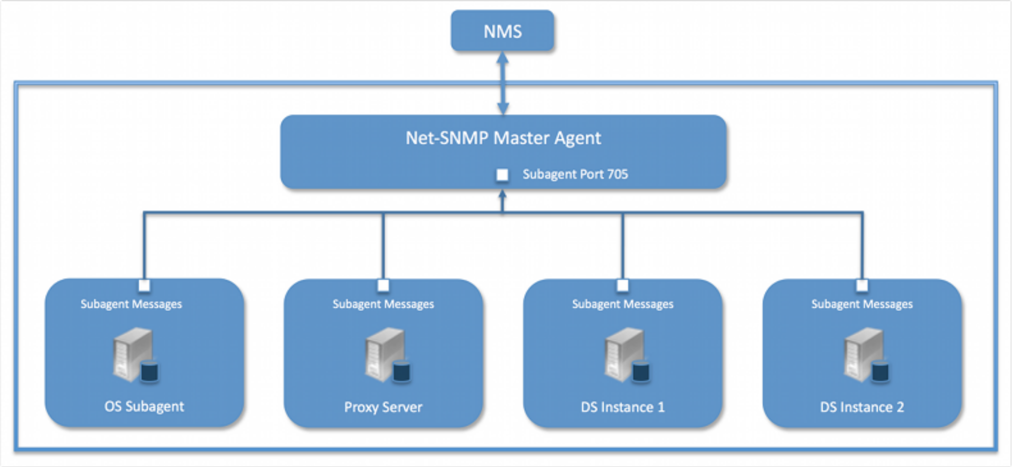


Figure 19: Example SNMP Deployment

One important note is that the PingDirectory Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note: The Directory Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

   ```
   $ bin/dsconfig set-alert-handler-prop \
     --handler-name "SNMP Subagent Alert Handler" --set enabled:true
   ```
2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

   ```
   The SNMP sub-agent was unable to connect to the master
   agent at localhost/705: Timeout
   ```
3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

   ```
   master agentx agentXSocket tcp:localhost:705
   ```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.
4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

   ```
   trap2sink localhost public 162
   ```
5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

   ```
   rwuser initial
   createUser initial MD5 setup_passphrase DES
   ```
6. Run the following command to create the SNMPv3 user.

   ```
   snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
   localhost create snmpuser initial
   ```
7. Start the `snmpd` daemon and after a few seconds you should see the following message in the Directory Server error log:

   ```
   The SNMP subagent connected successfully to the master agent
   at localhost:705. The SNMP context name is host.example.com:389
   ```
8. Set up a trap client to see the alerts that are generated by the Directory Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

   ```
   authcommunity log, execute public
   ```
9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

   ```
   $ cp resource/mib/* /usr/share/snmp/mibs
   ```
10. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (-f), do not read any configuration files (-C) except the one specified with the -c option, print to standard output (-Lo), and then specify that debugging output should be turned on for the User-based Security Module (-Dusm). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
  -m all -M +/usr/share/snmp/mibs
```

11. Run the Net-SNMP client tools to test the feature. The following options are required: -v <SNMP version>, -u <user name>, -A <user password>, -l <security level>, -n <context name (instance name)> . The -m all option loads all MIBs in the default MIB directory in /usr/share/snmp/mibs so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389
 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n
 host.example.com:389 \
-m all localhost systemStatus
```

MIBS

The Directory Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Monitoring with the Administrative Console

Ping Identity has a Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.

4. Use the top level navigation dropdown and select 'Status.'
5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

The PingDirectory Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the Administrative Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The PingDirectory Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Server using the `dsconfig` tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

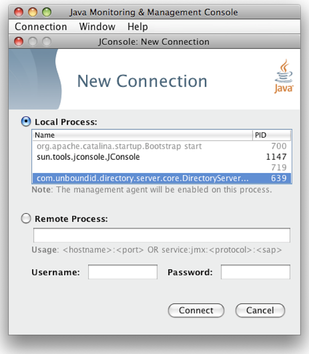
To Run JConsole

1. Use `JConsole` to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: `jconsole PID`. Or you can run JConsole remotely using: `jconsole hostname:port`.

   ```
   $ jconsole
   ```

   Note: If SSL is configured on the JMX Connection Handler, you must specify the Directory Server jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

   ```
   $ jconsole \
     -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
     -J-Djavax.net.ssl.trustStorePassword=secret \
     -J-Djava.class.path=$SERVER_ROOT/lib/PingDirectory.jar:/Library/Java/
   JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/jconsole.jar
   ```
2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.



● Local Process:

Name	PID
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1147
	719
com.unboundid.directory.server.core.DirectoryServer...	639

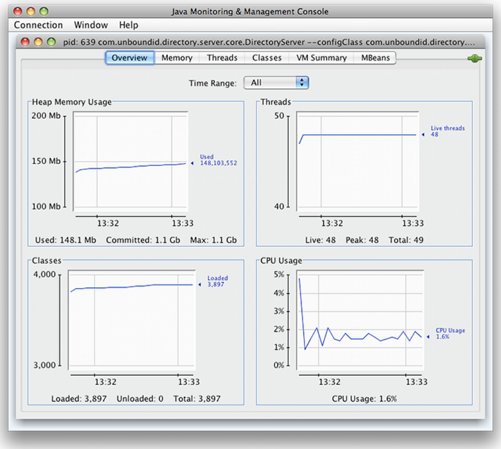
Note: The management agent will be enabled on this process.

○ Remote Process:

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: Password:

3. Review the resource monitoring information.



pid: 639 com.unboundid.directory.server.core.DirectoryServer --configClass com.unboundid.directory....

Time Range: All

Used: 148.1 Mb Committed: 1.1 Gb Max: 1.1 Gb

Live: 48 Peak: 48 Total: 49

Loaded: 3,897 Unloaded: 0 Total: 3,897

CPU Usage: 1.6%

Monitoring the Directory Server Using JConsole

You can set up JConsole to monitor the Directory Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscibe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Server using JConsole

1. Start the Directory Server.

```
$ bin/start-server
```

2. Enable the JMX Connection handler using the `dsconfig` tool. The handler is disabled by default. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

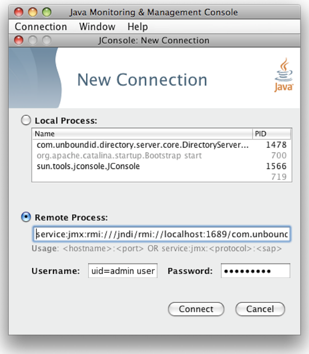
```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
```

```
ds-privilege-name: jmx-notify
```

4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



○ Local Process:

Name	PID
com.unboundid.directory.server.core.DirectoryServer...	1478
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1566
	719

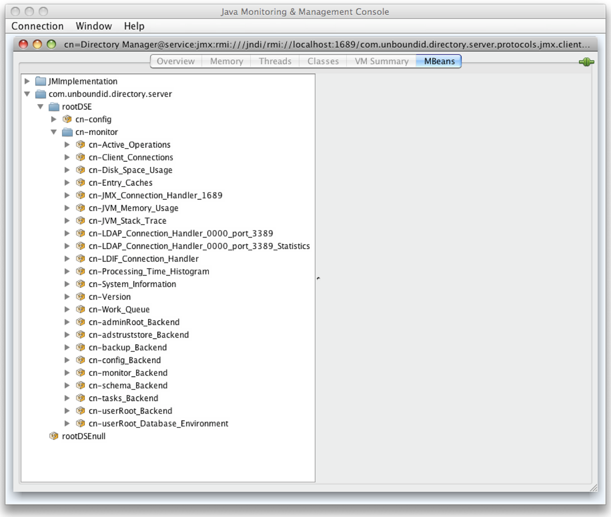
● Remote Process:

service:jmx:rmi:///jndi/rmi://localhost:1689/com.unbounc

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: uid=admin user Password: •••••••••

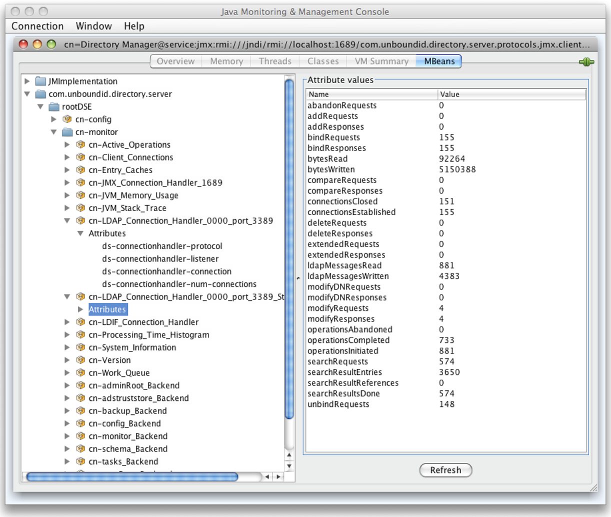
6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.



cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...

```
JMImplementation
com.unboundid.directory.server
  rootDSE
    cn-config
    cn-monitor
      cn-Active_Operations
      cn-Client_Connections
      cn-Disk_Space_Usage
      cn-Entry_Caches
      cn-JMX_Connection_Handler_1689
      cn-JVM_Memory_Usage
      cn-JVM_Stack_Trace
      cn-LDAP_Connection_Handler_0000_port_3389
      cn-LDAP_Connection_Handler_0000_port_3389_Statistics
      cn-LDIF_Connection_Handler
      cn-Processing_Time_Histogram
      cn-System_Information
      cn-Version
      cn-Work_Queue
      cn-adminRoot_Backend
      cn-adstruststore_Backend
      cn-backup_Backend
      cn-config_Backend
      cn-monitor_Backend
      cn-schema_Backend
      cn-tasks_Backend
      cn-userRoot_Backend
      cn-userRoot_Database_Environment
  rootDSEnull
```

7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.


Java Monitoring & Management Console
Connection
Window
Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview
Memory
Threads
Classes
VM Summary
MBeans
JMImplementation
com.unboundid.directory.server
rootDSE
cn-config
cn-monitor
cn-Active_Operations
cn-Client_Connections
cn-Entry_Caches
cn-JMX_Connection_Handler_1689
cn-JVM_Memory_Usage
cn-JVM_Stack_Trace
cn-LDAP_Connection_Handler_0000_port_3389
Attributes
ds-connectionhandler-protocol
ds-connectionhandler-listener
ds-connectionhandler-connection
ds-connectionhandler-num-connections
cn-LDAP_Connection_Handler_0000_port_3389_St
Attributes
cn-LDIF_Connection_Handler
cn-Processing_Time_Histogram
cn-System_Information
cn-Version
cn-Work_Queue
cn-adminRoot_Backend
cn-adstruststore_Backend
cn-backup_Backend
cn-config_Backend
cn-monitor_Backend
cn-schema_Backend
cn-tasks_Backend
Attribute values
Refresh

Name	Value
abandonRequests	0
addRequests	0
addResponses	0
bindRequests	155
bindResponses	155
bytesRead	92264
bytesWritten	5150388
compareRequests	0
compareResponses	0
connectionsClosed	151
connectionsEstablished	155
deleteRequests	0
deleteResponses	0
extendedRequests	0
extendedResponses	0
ldapMessagesRead	881
ldapMessagesWritten	4383
modifyDNRequests	0
modifyDNResponses	0
modifyRequests	4
modifyResponses	4
operationsAbandoned	0
operationsCompleted	733
operationsInitiated	881
searchRequests	574
searchResultEntries	3650
searchResultReferences	0
searchResultsDone	574
unbindRequests	148

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " +
 String.valueOf(value));
      }
     }
     System.out.println();
  }
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring over LDAP

The PingDirectory Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Profiling Server Performance Using the Stats Logger

The Directory Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Directory Server ships with the built-in "Stats Logger' disabled. To enable it using the `dsconfig` tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select .

1. Run `dsconfig` in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

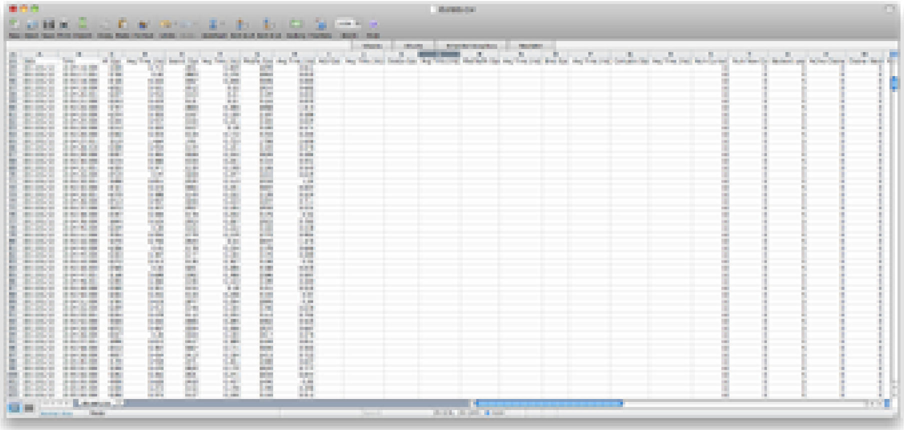
```
$ bin/dsconfig
```

2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plug-in.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

```
>>>> Configure the properties of the Stats Logger Plugin

Property                Value(s)
------------------------------------------------------------------------
1)   description        Logs performance stats to a log file
                        periodically.
2)   enabled                     false
3)   local-db-backend-info       basic
4)   replication-info            basic
5)   entry-cache-info            -
6)   host-info                   -
7)   included-ldap-application   If per-application LDAP stats is enabled,
                                 then stats will be included for all
                                 applications.
8)   log-interval                1 s
9)   collection-interval         200 ms
10)  suppress-if-idle            true
11)  header-prefix-per-column    false
12)  empty-instead-of-zero       true
13)  lines-between-header        50
14)  included-ldap-stat          active-operations, num-connections,
                                 op-count-and-latency, work-queue
15)  included-resource-stat      memory-utilization
16)  histogram-format            count
17)  histogram-op-type           all
18)  per-application-ldap-stats  aggregate-only
19)  ldap-changelog-info         -
20)  gauge-info                  none
21)  log-file                    logs/dsstats.csv
```

```
22)   log-file-permissions         640
23)   append                       true
24)   rotation-policy              Fixed Time Rotation Policy, Size Limit
                                   Rotation Policy
25)   retention-policy             File Count Retention Policy

?)    help
f)    finish - apply any changes to the Periodic Stats Logger Plugin
a)    hide advanced properties of the Periodic Stats Logger Plugin
d)    display the equivalent dsconfig command lines to either re-create this
            object or only to apply pending changes
b)    back
q)    quit

Enter choice [b]:
```

7. Run the Directory Server. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate` --help to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in To Enable the Stats Logger.
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

👉 **Note:** Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

Content-Reference to:

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. On the Directory Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.

11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                     Value(s)
         ---------------------------------------------------------------
    1)   description                  -
    2)   enabled                      true
    3)   monitor-objectclass          ds-memory-usage-monitor-entry
    4)   include-filter               -
    5)   attribute-to-log             total-bytes-used-by-memory-consumers
    6)   column-name                  Memory Consumer Total (GB)
    7)   statistic-type               raw
    8)   header-prefix                -
    9)   header-prefix-attribute      -
    10)  regex-pattern                -
    11)  regex-replacement            -
    12)  divide-value-by              1073741824
    13)  divide-value-by-attribute    -
    14)  decimal-format               #.##
    15)  non-zero-implies-not-idle    false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

Content-Reference to:

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

Chapter

22

Managing Notifications and Alerts

Topics:

The PingDirectory Server provides delivery mechanisms for account status notifications and administrative alerts using SMTP, JMX, or SNMP in addition to standard error logging. Alerts and events reflect state changes within the server that may be of interest to a user or monitoring service. Notifications are typically the delivery of an alert or event to a user or monitoring service. Account status notifications are only delivered to the account owner notifying a change in state in the account.

This chapter presents the following topics:

Working with Account Status Notifications

The PingDirectory Server supports notification handlers that can be used to notify users and/or administrators of significant changes related to password policy state for user entries. The following two notification handlers are available:

- **Error Log Account Status Notification Handler**. Enabled by default. The handlers send alerts to the error log when an account event occurs.
- **SMTP Account Status Notification Handler**. Disabled by default. You can enable the SMTP Handler with the `dsconfig` command to send notifications to designated email addresses.

Account Status Notification Types

The handlers send alerts when one of the account status events described in the following table occurs during password policy processing.

Table 64: Account Status Notification Types

Account Status Notification Types	Description
account-disabled	Generates a notification whenever a user account is disabled by an administrator.
account-enabled	Generates a notification whenever a user account is enabled by an administrator.
account-expired	Generates a notification whenever a user authentication attempt fails because the account has expired.
account-idle-locked	Generates a notification whenever a user authentication attempt fails because the account has been locked after idling for too long.
account-permanently-locked	Generates a notification whenever a user account is permanently locked (requiring administrative action to unlock the account) after too many failed attempts.
account-reset-locked	Generates a notification whenever an authentication attempt fails because the user account is locked because the user failed to change a password within the required interval that was reset by an administrator.
account-temporarily-locked	Generates a notification whenever a user account is temporarily locked after too many failed attempts.
account-unlocked	Generates a notification whenever a user account is unlocked by an administrator.
password-changed	Generates a notification whenever a user changes his or her own password.
password-expired	Generates a notification whenever a user authentication fails because the password has expired.
password-expiring	Generates a notification the first time that a password expiration warning is encountered for a user password.
password-reset	Generates a notification whenever a user's password is reset by an administrator.

Working with the Error Log Account Status Notification Handler

The Error Log Account Status Notification Handler is enabled by default and sends alerts when one of the account status events occur.

To Disable the Error Log Account Status Notification Handler

- Use the `dsconfig` tool to disable the Error Log Handler. You can view the log at `logs/error`.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  ---handler-name "Error Log Handler" --set enabled:false
```

To Remove a Notification Type from the Error Log Handler

- While not recommended, if you want to remove an account status notification type, use the `dsconfig` tool with the --remove option.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  --handler-name "Error Log Handler" \
  --remove account-status-notification-type: password-reset
```

Working with the SMTP Account Status Notification Handler

You can enable account status notifications to be sent to designated email addresses of end users, administrators, or both through an outgoing SMTP server. The email message is automatically generated from template files that contain the text to use in the message body. For example, the message subject for the account-disabled event is:

```
account-disabled: Your directory account has been disabled.
```

The message templates are located in the `config/messages` directory. The typical message body template is as follows:

```
Your directory account has been disabled.

For further assistance, please contact a server administrator.
```

By default, the sender address is notifications@example.com, but you can configure your own address.

Before you enable the SMTP Account Status Notification Handler, you must configure the Directory Server to use at least one mail server as shown below. You can configure an SMTP server using `dsconfig` and the set-global-configuration-prop option.

To Configure the SMTP Server

1. Use `dsconfig` to configure a simple mail server.

```
$ bin/dsconfig create-external-server --server-name smtp1 \
  --type smtp --set server-host-name:smtp.example.com
```

2. Use `dsconfig` to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  --set smtp-server:smtp1
```

To Configure a StartTLS Connection to the SMTP Server

1. Use `dsconfig` to configure a StartTLS connection to the server.

```
$ bin/dsconfig create-external-server \
  --server-name myTLSServer --type smtp \
  --set server-host-name:tls.smtp.example.com \
  --set server-port:587 \
  --set smtp-security:starttls \
  --set user-name:MyAccountName \
  --set password:AAD5yZ+DjvwiYkBSMer6GQ6B3szQ6gSSBjA=
```

2. Use `dsconfig` to configure a newly-created SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  -set smtp-server:myTLSServer
```

To Configure an SSL Connection to the SMTP Server

1. Use `dsconfig` to create an external SMTP server using SSL.

```
$ bin/dsconfig create-external-server \
  --server-name ssl.smtp.example.com \
  --type smtp --set server-host-name:stmp.gmail.com \
  --set server-port:465 \
  --set smtp-security:ssl \
  --set ’username:my.name@example.com’ \
  --set password:xxxxxx --set "smtp-timeout:10s"
```

2. Use `dsconfig` to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  --set smtp-server:ssl.smtp.example.com
```

To Enable the SMTP Account Status Notification Handler

- Use `dsconfig` to enable the SMTP account status notification handler.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  --handler-name "SMTP Handler" --set enabled:true \
  --set "recipient-address:admin@example.com" \
  --set "sender-address:acct-status-notifications@example.com
```

To View the Account Status Notification Handlers

- After you have enabled the SMTP server, view the list of account status notification handlers using `dsconfig`.

```
$ bin/dsconfig list-account-status-notification-handlers
```

```
Account Status Notification Handler : Type    : enabled
------------------------------------:-----------:--------
Error Log Handler                 : error-log : true
SMTP Handler                      : smtp    : true
```

Associating Account Status Notification Handlers with Password Policies

To generate notifications whenever appropriate password policy state changes occur in the server, the password policy that governs the entry being updated must be configured to use one or more account status notification handlers. By default, password policies are not configured with any such handlers, and therefore, no account status notifications will be generated.

The set of account status notification handlers that should be in use for a password policy is controlled by the `account-status-notification-handler` property for that password policy. It can be configured using `dsconfig` or the Administrative Console. For example, the following change updates the default password policy, so that the error log account status notification handler will be invoked for any appropriate password policy state changes for entries governed by the default password policy:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "account-status-notification-handler:Error Log Handler"
```

Working with Administrative Alert Handlers

The PingDirectory Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler**. Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Directory Server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler**. Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. Ping Identity uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler**. Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler**. Provides alert handler implementations created in third-party code using the Server SDK.

Administrative Alert Types

If enabled, the Directory Server can generate administrative alerts when the events occur. A full listing of system alerts and their severity is available in `<server-root>/docs/admin-alerts-list.csv`

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the `dsconfig` tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using `dsconfig` in interactive command-line mode, which is visible on the "Standard" object menu.

To Configure the JMX Connection Handler

1. Use `dsconfig` to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the `ldapmodify` tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
```

```
givenName: JMX
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

To Configure the JMX Alert Handler

- Use `dsconfig` to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the `dsconfig` tool.

Configuring the SMTP Alert Handler

- Use the `dsconfig` tool to configure the SMTP Alert Handler.

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the `dsconfig` tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP2c. For more information on SNMP, see Monitoring Using SNMP.

To Configure the SNMP Subagent Alert Handler

- Use `dsconfig` to configure the SNMP subagent alert handler. The `server-host-name` is the address of the system running the SNMP subagent. The `server-port` is the port number on which the subagent is running. The `community-name` is the name of the SNMP community that is used for the traps.

  The Directory Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

Working with the Alerts Backend

The Directory Server stores recently generated admin alerts in an Alerts Backend under the `cn=alerts` branch. The backend makes it possible to obtain admin alert information over LDAP for use with remote monitoring. The backend's primary job is to process search operations for alerts. It does not support add, modify, or modify DN operations of entries in the `cn=alerts` backend.

The alerts persist on disk in the `config/alerts.ldif` file so that they can survive server restarts. By default, the alerts remain on disk for seven days before being removed. However, administrators can configure the number of days for alert retention using the `dsconfig` tool. The administrative alerts of Warning level or worse that have occurred in the last 48 hours are viewable from the output of the status command-line tool and in the Administrative Console.

To View Information in the Alerts Backend

- The following uses `ldapsearch` to view the admin alerts.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alerts "(objectclass=ds-admin-alert)"
```

```
dn: ds-alert-id=3d1857a2-e8cf-4e80-ac0e-ba933be59eca,cn=alerts
objectClass: top
objectClass: ds-admin-alert
ds-alert-id: 3d1857a2-e8cf-4e80-ac0e-ba933be59eca
ds-alert-type: server-started
ds-alert-severity: info
ds-alert-type-oid: 1.3.6.1.4.1.32473.2.11.33
ds-alert-time: 20110126041442.622Z
ds-alert-generator: com.unboundid.directory.server.core.directory.server
ds-alert-message: The Directory Server has started successfully
```

To Modify the Alert Retention Time

1. Use `dsconfig` to change the maximum time information about generated admin alerts is retained in the Alerts backend. After this time, the information gets purged from the Directory Server. The minimum retention time is 0 milliseconds, which immediately purges the alert information.

```
$ bin/dsconfig set-backend-prop --backend-name "alerts" \
  --set "alert-retention-time: 2 weeks"
```

2. View the property using `dsconfig`.

```
$ bin/dsconfig get-backend-prop --backend-name "alerts" \
  --property alert-retention-time
```

```
Property : Value(s)
---------------------:---------
alert-retention-time : 2 w
```

To Configure Duplicate Alert Suppression

- Use `dsconfig` to configure the maximum number of times an alert is generated within a particular timeframe for the same condition. The `duplicate-alert-time-limit` property specifies the length of time that must pass before duplicate messages are sent over the administrative alert framework. The `duplicate-alert-limit` property specifies the maximum number of duplicate alert messages should be sent over the administrative alert framework in the time limit specified in the `duplicate-alert-time-limit` property.

```
$ bin/dsconfig set-global-configuration-prop \
  --set duplicate-alert-limit:2 \
  --set "duplicate-alert-time-limit:3 minutes"
```

Working with Alarms, Alerts, and Gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server unavailable' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and Administrative Console. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The Directory Server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

The Directory Server is compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when the Condition and Resource properties are the same. The Condition corresponds to the Summary column in the `admin-alerts-list.csv` file.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms.

To View Information in the Alarms Backend

- The following uses `ldapsearch` to view alarms. The following displays the listing for the CPU usage alarm.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alarms "(objectclass=ds-admin-alarm)"
```

```
dn: ds-alarm-id=CPU Usage (Percent)-Host System,cn=alarms
dn: ds-alarm-id=CPU Usage (Percent)-Host System,cn=alarms
objectClass: top
objectClass: ds-admin-alarm
ds-alarm-id: CPU Usage (Percent)-Host System
ds-alarm-condition: CPU Usage (Percent)
ds-alarm-specific-resource: Host System
ds-alarm-severity: CRITICAL
ds-alarm-previous-severity: CRITICAL
ds-alarm-details: Gauge CPU Usage (Percent) for Host System
       has value 99, having had a value of 83.13 in the
       previous interval. The severity is critical, having
       assumed this severity Thu Sep 25 10:24:20 CDT 2014
       when the value crossed threshold 80
```

```
ds-alarm-additional-text: If CPU use is high, check the server's current
 workload
      and other processes on this system and make any needed adjustments.
 Reducing
      the load on the system will lead to better response times
ds-alarm-start-time: 20140925152420.004Z
ds-alarm-critical-last-time: 20140925152420.004Z
ds-alarm-critical-total-duration-millis: 0
```

Testing Alerts and Alarms

After alarms and alert handlers are configured, verify that the server takes the appropriate action when an alarm state changes by manually increasing the severity of a gauge. Alarms and alerts can be verified with the `status` tool.

To Test Alarms and Alerts

1. Configure a gauge with `dsconfig` and set the `override-severity` property to critical. The following example uses the CPU Usage (Percent) gauge.

```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

2. Run the `status` tool to verify that an alarm was generated with corresponding alerts. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status

                        --- Administrative Alerts ---
 Severity : Time             : Message
 ---------:------------------:------------------------------------------------------
 Info     : 11/Aug/2014      : A configuration change has been made in the
          : 15:48:46 -0500   : Directory Server:
          :                  : [11/Aug/2014:15:48:46.054 -0500]
          :                  : conn=17 op=73 dn='cn=Directory Manager,cn=Root
          :                  : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                  : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                  :  --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                  : --set warning-value:-1'
 Info     : 11/Aug/2014      : A configuration change has been made in the
          :  15:47:32 -0500  : Directory Server: [11/Aug/2014:15:47:32.547
 -0500]
          :                  : conn=4 op=196 dn='cn=Directory Manager,cn=Root
          :                  : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                  : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                  : --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                  :  --set warning-value:0'
 Error    : 11/Aug/2014      : Alarm [CPU Usage (Percent). Gauge CPU Usage
 (Percent)
          :  15:41:00 -0500  : for Host System has
          :                  : a current value of '18.583333333333332'.
          :                  : The severity is currently OVERRIDDEN in the
          :                  : Gauge's configuration to 'CRITICAL'.
          :                  : The actual severity is: The severity is
          :                  : currently 'NORMAL', having assumed this
 severity
```

```
          :                 : Mon Aug 11 15:41:00 CDT 2014. If CPU use is
 high,
          :                 : check the server's current workload and make
 any
          :                 : needed adjustments. Reducing the load on the
 system
          :                 : will lead to better response times.
          :                 : Resource='Host System']
          :                 : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
 hours
 Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
                         --- Alarms ---
 Severity : Severity Start : Condition : Resource    : Details
          : Time           :           :             :
 ---------:----------------:-----------:-------------:------------------------------
 Critical : 11/Aug/2014    : CPU Usage : Host System : Gauge CPU Usage
 (Percent) for
          : 15:41:00 -0500 : (Percent) :             : Host System
          :                :           :             : has a current value
 of
          :                :           :             : '18.785714285714285'.
          :                :           :             : The severity is
 currently
          :                :           :             : 'CRITICAL', having
 assumed
          :                :           :             : this severity Mon Aug
 11
          :                :           :             : 15:49:00 CDT 2014. If
 CPU use
          :                :           :             : is high, check the
 server's
          :                :           :             : current workload and
 make any
          :                :           :             : needed adjustments.
 Reducing
          :                :           :             : the load on the
 system will
          :                :           :             : lead to better
 response times
 Warning  : 11/Aug/2014    : Work Queue: Work Queue  : Gauge Work Queue Size
 (Number
          : 15:39:40 -0500 : Size      :             : of Requests) for Work
 Queue
          :                : (Number of:             : has a current value
 of '27'.
          :                : Requests) :             : The severity is
 currently
          :                :           :             : 'WARNING' having
 assumed this
          :                :           :             : severity Mon Aug 11
 15:48:50
          :                :           :             : CDT 2014. If all
 worker
          :                :           :             : threads are busy
 processing
          :                :           :             : other client
 requests, then
          :                :           :             : new requests that
 arrive will
          :                :           :             : be forced to wait in
 the work
```

```
          :                :           :             : queue until a worker
 thread
          :                :           :             : becomes available
Shown are alarms of severity [Warning,Minor,Major,Critical]
Use the --alarmSeverity option to filter this list
```

Indeterminate Alarms

Indeterminate alarms are raised for a server condition for which a severity cannot be determined. In most cases these alarms are benign and do not issue alerts nor appear in the output of the `status` tool or Administrative Console by default. These alarms are usually caused by an enabled gauge that is intended to measure an aspect of the server that is not currently enabled. For example, gauges intended to monitor metrics related to replication may produce indeterminate alarms if a Directory Server is not currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There may be messages that can help determine the issue. The following is sample output from the `status` tool run with the –`alarmSeverity=indeterminate` option:

```
                        --- Alarms ---
Severity     : Severity Start : Condition      : Resource   : Details
             : Time           :                :            :
-------------:----------------:----------------:------------:------------------------
Normal       : 26/Aug/2014    : Startup Begun  : cn=config  : The Directory
 Server
             : 14:16:29 -0500 :                :            : is starting.
             :                :                :            :
Indeterminate: 26/Aug/2014    : Replication    : not        : The value of
 gauge
             : 14:16:40 -0500 : Latency        : available  : Replication
 Latency
             :                : (Milliseconds) :            : (Milliseconds)
 could not
             :                :                :            : be determined.
 The
             :                :                :            : severity is
 INDETERMINATE,
             :                :                :            : having assumed
 this
             :                :                :            : severity Tue Aug
 26
             :                :                :            : 14:17:10 CDT
 2014.
```

The following is an indeterminate alarm for the Replication Latency (Milliseconds) gauge. The following is a sample search of the monitor backend for this gauge’s entry. The result is an error message may explain the indeterminate severity:

```
# ldapsearch -w password --baseDN "cn=monitor"  \
-D"cn=directory manager" gauge-name="Replication Latency (Milliseconds)"

dn: cn=Gauge Replication Latency (Milliseconds),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-numeric-gauge-monitor-entry
objectClass: ds-gauge-monitor-entry
objectClass: extensibleObject
cn:          Gauge Replication Latency (Milliseconds)
gauge-name:  Replication Latency (Milliseconds)
resource:
severity:    indeterminate
summary:     The value of gauge Replication Latency (Milliseconds) could not
             be determined. The severity is INDETERMINATE, having assumed
```

```
              this severity Tue Aug 26 15:42:40 CDT 2014
error-message: No entries were found under cn=monitor having object
               class ds-replica-monitor-entry
              …
```

Chapter

23

Managing the SCIM Servlet Extension

Topics:

The PingDirectory Server provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS.

This section describes fundamental SCIM concepts and provides information on configuring SCIM on your server.

Overview of SCIM Fundamentals

Understanding the basic concepts of SCIM can help you use the SCIM extension to meet the your deployment needs. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension implements the 1.1 version of the SCIM specification. Familiarize yourself with this specification to help you understand and make efficient use of the SCIM extension and the SCIM SDK. The SCIM specifications are located on the Simplecloud website.

Summary of SCIM Protocol Support

PingData implements all required features of the SCIM protocol and most optional features. The following table describes SCIM features and whether they are supported.

Table 65: SCIM Protocol Support

SCIM Feature	Supported
Etags	Yes
JSON	Yes
XML*	Yes
Authentication/Authorization	Yes, via HTTP basic authentication or OAuth 2.0 bearer tokens
Service Provider Configuration	Yes
Schema	Yes
User resources	Yes
Group resources	Yes
User-defined resources	Yes
Resource retrieval via GET	Yes
List/query resources	Yes
Query filtering*	Yes
Query result sorting*	Yes
Query result pagination*	Yes (Directory Server, not Directory Proxy Server)
Resource updates via PUT	Yes
Partial resource updates via PATCH*	Yes
Resource deletes via DELETE	Yes
Resource versioning*	Yes (requires configuration for updated servers)
Bulk*	Yes
HTTP method overloading	Yes
Raw LDAP Endpoints**	Yes

* denotes an optional feature of the SCIM protocol.

** denotes a PingData extension to the basic SCIM functionality.

About the Identity Access API

The PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server support an extension to the SCIM standard called the Identity Access API. The Identity Access API provides an alternative to LDAP by supporting CRUD (create, read, update, and delete) operations to access directory server data over an HTTP connection.

SCIM and the Identity Access API are provided as a unified service through the SCIM HTTP Servlet Extension. The SCIM HTTP Servlet Extension can be configured to only enable core SCIM resources (e.g., 'Users' and 'Groups'), only LDAP object classes (e.g., `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there may be two representations with different schemas for any resources defined in the `scim-resources.xml` file: the SCIM representation and the raw LDAP representation. Likewise, because resources are exposed by an LDAP object class, and because these are hierarchical (e.g., `top` --> `person` --> `organizationalPerson` --> `inetOrgPerson`, etc.), a client application can access an entry in multiple ways due to the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the PingDirectory Server.

Configuring SCIM

This section discusses details about the PingData implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available on the Simplecloud website.

Creating Your Own SCIM Application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM SDK is available for download at https://github.com/pingidentity/scim.

Note: The value of a read-only SCIM attribute can be set by a POST operation if the SCIM attribute is a custom attribute in the `scim-resource.xml` config file, but not if the SCIM attribute is a core SCIM attribute.

Configuring the SCIM Servlet Extension

The Directory Server provides a default SCIM HTTP Servlet Extension that can be enabled and configured using a `dsconfig` batch script located in the `config` directory. The script runs a series of commands that enables an HTTPS Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension. There are additional optional configurations (e.g., changing the log format, enable `entryDN` virtual attribute and using VLV indexes) that you can make by altering the `dsconfig` batch script.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See Managing the SCIM Schema for more information.

The following procedures show how to configure SCIM on the server. The first example procedure shows the steps to manually configure SCIM without using the script. The second example procedure uses the `dsconfig` batch script to configure SCIM.

To Configure SCIM Manually

The following example procedure assumes that you have configured the Directory Server using the default settings, which means that SSL and the HTTPS Connection Handler have not been configured. The example also shows the `dsconfig` non-interactive commands. You can easily use the `dsconfig` interactive commands, which uses a menu-driven interface. If you use the `dsconfig` interactive, you must change to the Standard or Advanced object menu to access many of these configuration settings.

1. Set up your certificates. Follow the examples shown in the section *Managing Certificates*. You should have a keystore and truststore set up in the `config` directory. Make sure that the keystore.pin and truststore.pin are set.
2. Enable the key manager provider. The key manager provider accesses the certificate during the SSL handshaking process. If running `dsconfig` interactive, open the main menu, select "Key Manager Provider" -> "View and edit an existing Key Manager Provider" -> "JKS" (or the type of certificate you are working with) -> "enabled" and then set the value to "true". Click "finish" to save the setting.

```
$ bin/dsconfig create-key-manager-provider --provider-name JKS \
  --type file-based --set enabled:true --set key-store-file:config/keystore
 \
  --set key-store-type:JKS --set key-store-pin-file:config/keystore.pin
```

3. Enable the trust manager provider. The trust manager provider determines if a presented certificate can be trusted. If running `dsconfig` interactive, open the main menu, select "Trust Manager Provider" -> "View and edit an existing Trust Manager Provider" -> "JKS" (or the type of certificate you are working with) -> "enabled" and then set the value to "true". Click "finish" to save the setting.

```
$ bin/dsconfig create-trust-manager-provider --provider-name JKS \
  --type file-based --set enabled:true --set trust-store-file:config/
truststore \
  --set trust-store-type:JKS --set trust-store-pin-file:config/
truststore.pin
```

4. Configure the HTTPS Connection Handler. The command enables the connection handler, specifies the SCIM HTTP servlet extension and sets the listen port to a port of your choice, in this example, use 8443. The command also specifies the type of key manager and trust manager providers and sets the log publisher to "HTTP Detailed Access." If running `dsconfig` interactive, open the main menu, select "Connection Handler" -> "View and edit an existing Connection Handler" -> "HTTPS Connection Handler". Change the parameters to match your setup, and then, click "finish" to save the setting.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "HTTPS Connection Handler" \
  --type http --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set http-servlet-extension:SCIM \
  --set "http-operation-log-publisher:HTTP Detailed Access" \
  --set key-manager-provider:JKS --set trust-manager-provider:JKS
```

5. Add access controls to allow access to LDAP controls used by the SCIM Servlet Extension. These controls are the Post-Read Request Control (1.3.6.1.1.13.2), Server-Side Sort Request Control (1.2.840.113556.1.4.473), Simple Paged Results Control (1.2.840.113556.1.4.319), and Virtual List View Request Control (2.16.840.1.113730.3.4.9). We recommend using the following command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473
 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9")
    (version 3.0;acl "Authenticated access to controls used by the SCIM
 servlet
     extension"; allow (all) userdn="ldap:///all";)'
```

6. Add access controls to allow read access to operational attributes used by the SCIM Servlet Extension. We recommend using the following non-interactive command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id
 ||
    createTimestamp || modifyTimestamp")
    (version 3.0;acl "Authenticated read access to operational attributes \
    used by the SCIM servlet extension"; allow (read,search,compare)
    userdn="ldap:///all";)'
```

7. Optional. The SCIM HTTP Connection Handler automatically uses a detailed HTTP log publisher, which is implemented in a proprietary format. If you need a standard W3C common log format publisher, enter the following command. If running `dsconfig` interactive, open the main menu, select "Log Publisher" -> "Create a new Log Publisher" -> "new Log Publisher created from scratch" -> "File Based Access Log Publisher", enter the parameters to match your setup, and then, click "finish" to save the setting. Go back to the main menu, select "Connection Handler" -> "HTTPS Connection Handler", and then add "HTTP Common Access" to the `http-operation-log-publisher` property. Click "finish" to save the setting.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Common Access" \
  --type common-log-file-http-operation --set enabled:true \
  --set log-file:logs/http-common-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"

$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add "http-operation-log-publisher:HTTP Common Access"
```

8. Optional. To support searching or filtering by DN using the Identity Access API, you can enable the entryDN virtual attribute. If running `dsconfig` interactive, open the main menu, select "Virtual Attribute" -> "View and edit an existing Virtual Attribute" -> "Entry DN", and then change the enabled property to "true". Click "finish" to save the setting.

```
$ bin/dsconfig set-virtual-attribute-prop --name entryDN --set enabled:true
```

9. Optional. To support pagination, create some Virtual List View indexes. If running `dsconfig` interactive, open the main menu, select "Local DB VLV Index" -> "Create a new Local DB VLV Index" and then enter the properties needed for your setup. Click "finish" to save the setting. Repeat again for the "ascending-sn" index. Then, run the `rebuild-index` command to let the VLV Indexes take effect.

```
$ bin/dsconfig create-local-db-vlv-index --backend-name userRoot \
--index-name ascending-uid --set base-dn:dc=example,dc=com \
--set scope:whole-subtree --set "filter:(objectclass=inetorgperson)" \
--set "sort-order:+uid"

$ bin/dsconfig create-local-db-vlv-index --backend-name userRoot \
--index-name ascending-sn --set base-dn:dc=example,dc=com \
--set scope:whole-subtree --set "filter:(objectclass=inetorgperson)" \
--set "sort-order:+sn"

$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.ascending-uid \
--index vlv.ascending-sn
```

To Enable Resource Versioning

Resource versioning is enabled by default in new installations. Upgraded servers that had SCIM enabled need additional configuration to enable resource versioning.

1. Enable the `ds-entry-checksum` virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
    --name ds-entry-checksum \
    --set enabled:true
```

2. Remove any existing access controls required by SCIM for read access to operational attributes:

```
$ bin/dsconfig set-access-control-handler-prop \
    --remove 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-
unique-id || createTimestamp || ds-create-time || modifyTimestamp || ds-
update-time")(version 3.0;acl "Authenticated read access to operational
 attributes used by the SCIM servlet extension"; allow (read,search,compare)
 userdn="ldap:///all"'
```

3. Add new access controls required by SCIM for read access to operational attributes with the addition of the `ds-entry-checksum`:

```
$ bin/dsconfig set-access-control-handler-prop \
    --add  'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-
id || createTimestamp || ds-create-time || modifyTimestamp || ds-update-
time || ds-entry-checksum")(version 3.0;acl "Authenticated read access
 to operational attributes used by the SCIM servlet extension"; allow
 (read,search,compare) userdn="ldap:///all"'
```

4. Enable SCIM resource versioning using the entry checksum virtual attribute:

```
$ bin/dsconfig set-http-servlet-extension-prop \
    --extension-name SCIM \
    --set entity-tag-ldap-attribute:ds-entry-checksum
```

   If enabled, the value of the `ds-entry-checksum` attribute is returned as the `ETag` header value when accessing the resource through SCIM, and is checked against the `If-Match` header when updating the resource. When accessing the resource through LDAP, use the `ds-entry-checksum` attribute instead.

To Configure the SCIM Servlet Extension using the Batch Script

The following example procedure assumes that you have set up your certificates, keystore, and truststore

1. Open the `<server-root>/config/scim-config-ds.dsconfig` script in a text editor.
2. For the optional elements (W3C common log, filtering by DN, and VLV Indexes, remove the comment (#) symbol on the `dsconfig` commands. Save the file when finished editing.
3. To enable the SCIM servlet extension, run the `dsconfig` batch file. Remember to include the bind parameters.

```
$ bin/dsconfig --batch-file config/scim-config-ds.dsconfig
```

SCIM Servlet Extension Authentication

The SCIM servlet supports authentication using either the HTTP Basic authentication scheme, or OAuth 2.0 bearer tokens. When authenticating using HTTP Basic authentication, the SCIM servlet attempts to correlate the username component of the Authorization header to a DN in the Directory Server. If the username value cannot be parsed directly as a DN, it is correlated to a DN using an Identity Mapper. The DN is then used in a simple bind request to verify the password.

In deployments that use an OAuth authorization server, the SCIM extension can be configured to authenticate requests using OAuth bearer tokens. The SCIM extension supports authentication with OAuth 2.0 bearer tokens (per RFC 6750) using an OAuth Token Handler Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, PingData provides the token handler API to decode incoming bearer tokens and extract or correlate associated authorization DNs.

Neither HTTP Basic authentication nor OAuth 2.0 bearer token authentication are secure unless SSL is used to encrypt the HTTP traffic.

To Configure Basic Authentication Using an Identity Mapper

By default, the SCIM servlet is configured to use the Exact Match Identity Mapper, which matches against the `uid` attribute. In this example, an alternate Identity Mapper is created so that clients can authenticate using `cn` values.

1. Create a new Identity Mapper that uses a match attribute of cn.

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "CN Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:cn
```

2. Configure the SCIM servlet to use the new Identity Mapper.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set "identity-mapper:CN Identity Mapper"
```

To Enable OAuth Authentication

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the Server SDK documentation.

1. Install your OAuth token handler on the server using `dsconfig`.

```
$ bin/dsconfig create-oauth-token-handler \
  --handler-name ExampleOAuthTokenHandler \
  --type third-party \
  --set extension-
class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
```

2. Configure the SCIM servlet extension to use it as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set oauth-token-handler:ExampleOAuthTokenHandler
```

Verifying the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

To Verify the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

- Run `curl` to verify that the SCIM extension is running. The -k (or --insecure) option is used to turn off curl's verification of the server certificate, since the example Directory Server is using a self-signed certificate.

```
$ curl -u "cn=Directory Manager:password" \
-k "https://localhost:8443/scim/ServiceProviderConfigs"

{"schemas":["urn:scim:schemas:core:1.0"],"id":"urn:scim:schemas:core:1.0",
"patch":{"supported":true},"bulk":{"supported":true,"maxOperations":10000,
"maxPayloadSize":10485760},"filter":{"supported":true,"maxResults":100},
"changePassword":{"supported":true},"sort":{"supported":true},
"etag":{"supported":false},"authenticationSchemes":[{"name":"HttpBasic",
"description":"The HTTP Basic Access Authentication scheme. This scheme is
not considered to be a secure method of user authentication (unless used in
conjunction with some external secure system such as SSL), as the user
name and password are passed over the network as cleartext.","specUrl":
"http://www.ietf.org/rfc/rfc2617","documentationUrl":
```

```
"http://en.wikipedia.org/wiki/Basic_access_authentication"}]}
```

- If the user ID is a valid DN (such as `cn=Directory Manager`), the SCIM extension authenticates by binding to the Directory Server as that user. If the user ID is not a valid DN, the SCIM extension searches for an entry with that `uid` value, and binds to the server as that user. To verify authentication to the server as the user with the `uid` of `user.0`, run the following command:

```
$ curl -u "user.0:password" \
  -k "https://localhost:8443/scim/ServiceProviderConfigs"
```

Configuring Advanced SCIM Extension Features

The following sections show how to configure advanced SCIM servlet extension features, such as bulk operation implementation, mapping SCIM resource IDs, and transformations.

Managing the SCIM Schema

This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About SCIM Schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping LDAP Schema to SCIM Resource Schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

Note that LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). The easiest and most-correct way to handle this is to make sure that any attributes that may contain binary data are declared using "dataType=binary" in the `scim-resources.xml` file. Likewise, when using the Identity Access API make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes which may contain binary data. This will cause the server to automatically base64-encode the data before returning it to clients and will also make it predictable for clients because they can assume the data will always be base64-encoded.

However, it is still possible that attributes that are not declared as binary in the schema may contain binary data (or just data that is invalid in XML), and the server will always check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values which are found to include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the <resource> Element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A resource element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the <attribute> Element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as urn:scim:schemas:core:1.0. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is false.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:

  `simple`: specifies a simple, singular SCIM attribute.
  `complex`: specifies a complex, singular SCIM attribute.
  `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
  `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the <simple> Element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: binary, boolean, dateTime, decimal, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simple element contains the following XML element:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the <complex> Element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

About the <simpleMultivalued> Element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simpleMultiValued element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the <complexMultiValued> Element

A `complexMultiValued` element has the following XML attribute:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is address.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.
- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the <subAttribute> Element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: binary, boolean, dateTime, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the complexMultiValued element.

About the <canonicalValue> Element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the <mapping> Element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in the Mapping LDAP Entries to SCIM Using the SCIM-LDAP API section.

About the <subMapping> Element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. The available transformations are described in *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API*.

About the <LDAPSearch> Element

An `LDAPSearch` element contains the following XML elements in sequence:

- `baseDN`: a required element specifying one or more LDAP search base DNs to be used when querying for the SCIM resource.
- `filter`: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- `resourceIDMapping`: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN. Note The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

Note: The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

About the <resourceIDMapping> Element

The `resourceIDMapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `scim-consumer`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or Directory

Server, meaning that a value is automatically provided by the Directory Server (as would be the case if the mapped LDAP attribute is entryUUID).

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the <LDAPAdd> Element

An LDAPAdd element contains the following XML elements in sequence:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

About the <fixedAttribute> Element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The value merge indicates that the fixed values should be merged with the existing values. The value overwrite indicates that the existing values are to be overwritten by the fixed values. The value preserve indicates that no changes should be made. The default value is merge.

A `fixedAttribute` element contains one or more `fixedValue` XML element, which specify the fixed LDAP values.

Validating Updated SCIM Schema

The PingData SCIM extension is bundled with an XML Schema document, `resources.xsd`, which describes the structure of a `scim-resources.xml` resource configuration file. After updating the resource configuration file, you should confirm that its contents are well-formed and valid using a tool such as `xmllint`.

For example, you could validate your updated file as follows:

```
$ xmllint --noout --schema resources.xsd scim-resources.xml
scim-resources.xml validates
```

Mapping SCIM Resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. The `entryUUID` attribute, whose read-only value is assigned by the Directory Server, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership. The resource configuration allows the SCIM resource ID to be mapped to the LDAP entry DN. However, the entry DN does not meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN. Likewise, you can use the Identity Access API to change the value of an entry's RDN attribute, thereby triggering a MODDN operation.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Directory Server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin may be used by the Directory Server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see "Working with the UID Unique Attribute Plug-in" in the PingDirectory Server Administration Guide.

Note: Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the `<resourceIDMapping>` Element".

Using Pre-defined Transformations

Transformations are required to change SCIM data types to LDAP syntax values. The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation`. Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation`. Transforms SCIM dateTime data type values to LDAP Generalized Time syntax values and vice-versa.
- `com.unboundid.scim.ldap.PostalAddressTransformation`. Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the $ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.
- `com.unboundid.scim.ldap.TelephoneNumberTransformation`. Transforms LDAP Telephone Number syntax (E.123) to RFC3966 format and vice-versa.

You can also write your own transformations using the SCIM API described in the following section.

Mapping LDAP Entries to SCIM Using the SCIM-LDAP API

In addition to the SCIM SDK, PingData provides a library called SCIM-LDAP, which provides facilities for writing custom transformations and more advanced mapping.

You can add the SCIM-LDAP library to your project using the following dependency:

```
<dependency>
    <groupId>com.unboundid.product.scim</groupId>
    <artifactId>scim-ldap</artifactId>
    <version>1.5.0</version>
</dependency>
```

Create your custom transformation by extending the `com.unboundid.scim.ldap.Transformation` class. Place your custom transformation class in a jar file in the server's `lib` directory.

Note: The Identity Access API automatically maps LDAP attribute syntaxes to the appropriate SCIM attribute types. For example, an LDAP DirectoryString is automatically mapped to a SCIM string.

SCIM Authentication

SCIM requests to the LDAP endpoints will support HTTP Basic Authentication and OAuth2 Authentication using a bearer token. There is existing support for this feature in the Directory Server and the Directory Proxy Server using the OAuthTokenHandler API (i.e., via a Server SDK extension, which requires some technical work to implement).

Note that our implementation only supports the HTTP Authorization header for this purpose; we do not support the form-encoded body parameter or URI query parameter mechanisms for specifying the credentials or bearer token.

SCIM Logging

The Directory Server already provides a detailed HTTP log publisher to capture the SCIM and HTTP request details. To be able to correlate this data to the internal LDAP operations that are invoked behind the scenes, the Access Log Publisher will use "`origin=scim`" in access log messages that are generated by the SCIM servlet.

For example, you will see a message for operations invoked by replication:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191
origin="replication" dn="uid=user.3,ou=people,dc=example,dc=com"
```

Likewise for SCIM messages, you will see a message like this:

```
[30/Oct/2012:18:45:10.490 -0500] MODFIY REQUEST conn=-3 op=190 msgID=191
origin="scim" dn="uid=user.3,ou=people,dc=example,dc=com"
```

SCIM Monitoring

There are two facilities that can be used to monitor the SCIM activity in the server.

- **HTTPConnectionHandlerStatisticsMonitorProvider** -- Provides statistics straight about total and average active connections, requests per connection, connection duration, processing time, invocation count, etc.
- **SCIMServletMonitorProvider** -- Provides high level statistics about request methods (POST, PUT, GET, etc.), content types (JSON, XML), and response codes, for example, "`user-patch-404:26`".

The LDAP object class endpoints are treated as their own resource types, so that for requests using the Identity Access API, there will be statistics, such as `person-get-200` and `inetorgperson-post-401`.

Configuring the Identity Access API

Once you have run the `<server-root>/config/scim-config-ds.dsconfig` script, the resources defined in the `scim-resources.xml` will be available as well as the Identity Access API. However, to allow SCIM access to the raw LDAP data, you must set a combination of configuration properties on the SCIM Servlet Extension using the `dsconfig` tool.

- **include-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will be exposed. The object class used here will be the one that clients need to use when referencing Identity Access API resources. This property allows the special value "*" to allow all object classes. If "*" is used, then the SCIM servlet uses the same case used in the Directory Server LDAP Schema.
- **exclude-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will not be exposed. When this property is specified, all object classes will be exposed except those in this list.
- **include-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will be exposed. If specified, only entries under these base DNs will be accessible. No parent-child relationships in the DNs are allowed here.
- **exclude-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will not be exposed. If specified, entries under these base DNs will not be accessible. No parent-child relationships in the DNs are allowed here.

Using a combination of these properties, SCIM endpoints will be available for all included object classes, just as if they were SCIM Resources defined in the `scim-resources.xml` file.

To Configure the Identity Access API

1. Ensure that you have run the `scim-config-ds.dsconfig` script to configure the SCIM interface. Be sure to enable the entryDN virtual attribute. See the Configure SCIM section for more information.
2. Set a combination of properties to allow the SCIM clients access to the raw LDAP data: `include-ldap-objectclass`, `exclude-ldap-objectclass`, `include-ldap-base-dn`, or `exclude-ldap-base-dn`.

   ```
   $ bin/dsconfig set-http-servlet-extension-prop \
     --extension-name SCIM --set 'include-ldap-objectclass:*' \
     --set include-ldap-base-dn:ou=People,dc=example,dc=com
   ```

   The SCIM clients now have access to the raw LDAP data via LDAP object class-based resources as well as core SCIM resources as defined in the `scim.resource.xml` file.

To Disable Core SCIM Resources

1. Open the `config/scim-resources.xml` file, and comment out or remove the `<resource>` elements that you would like to disable.

2. Disable and re-enable the HTTP Connection Handler, or restart the server to make the changes take effect. In general, changing the `scim-resources.xml` file requires a HTTP Connection Handler restart or server restart.

   Note: When making other changes to the SCIM configuration by modifying the SCIM HTTP Servlet Extension using `dsconfig`, the changes take effect immediately without any restart required.

To Verify the Identity Access API Configuration

- Perform a curl request to verify the Identity Access API configuration.

```
$ curl -k -u "cn=directory manager:password" \
  -H "Accept: application/json" \
  "https://example.com/top/56c9fd6b-f870-35ef-9959-691c783b7318?
     attributes=entryDN,uid,givenName,sn,entryUUID"
     {"schemas":
["urn:scim:schemas:core:1.0","urn:unboundid:schemas:scim:ldap:1.0"],
      "id":"56c9fd6b-f870-35ef-9959-691c783b7318",
      "meta":{"lastModified":"2013-01-11T23:38:26.489Z",
      "location":"https://example.com:443/v1/top/56c9fd6b-
f870-35ef-9959-691c783b7318"},
      "urn:unboundid:schemas:scim:ldap:1.0":{"givenName":["Rufus"],"uid":
["user.1"],
      "sn":["Firefly"],"entryUUID":["56c9fd6b-f870-35ef-9959-691c783b7318"],
      "entrydn":"uid=user.1,ou=people,dc=example,dc=com"}}
```

Monitoring the SCIM Servlet Extension

The SCIM SDK provides a command-line tool, `scim-query-rate`, that measures the SCIM query performance for your extension. The SCIM extension also exposes monitoring information for each SCIM resource, such as the number of successful operations per request, the number of failed operations per request, the number of operations with XML or JSON to and from the client. Finally, the Directory Server automatically logs SCIM-initiated LDAP operations to the default File-based Access Logger. These operations will have an `origin='scim'` attribute to distinguish them from operations initiated by LDAP clients. You can also create custom logger or request criteria objects that can track incoming HTTP requests, which the SCIM extension rewrites as internal LDAP operations.

Testing SCIM Query Performance

You can use the `scim-query-rate` tool, provided in the SCIM SDK, to test query performance, by performing repeated resource queries against the SCIM server.

The `scim-query-rate` tool performs searches using a query filter or can request resources by ID. For example, you can test performance by using a filter to query randomly across a set of one million users with eight concurrent threads. The user resources returned to the client in this example is in XML format and includes the `userName` and `name` attributes.

```
scim-query-rate --hostname server.example.com --port 80 \
--authID admin --authPassword password --xml \
--filter 'userName eq "user.[1-1000000]"' --attribute userName \
--attribute name --numThreads 8
```

You can request resources by specifying a resource ID pattern using the --resourceID argument as follows:

```
scim-query-rate --hostname server.example.com --port 443 \
--authID admin --authPassword password --useSSL --trustAll\
--resourceName User \
--resourceID 'uid=user.[1-150000],ou=people,dc=example,dc=com'
```

The `scim-query-rate` tool reports the error "`java.net.SocketException: Too many open files`" if the open file limit is too low. You can increase the open file limit to increase the number of file descriptors.

Monitoring Resources Using the SCIM Extension

The monitor provider exposes the following information for each resource:

Number of successful operations per request type (such as GET, PUT, and POST).

Number of failed operations and their error codes per request type.

Number of operations with XML or JSON from client.

Number of operations that sent XML or JSON to client.

In addition to the information about the user-defined resources, monitoring information is also generated for the schema, service provider configuration, and monitor resources. The attributes of the monitor entry are formatted as follows:

```
{resource name}-resource-{request type}-{successful or error status code}
```

You can search for one of these monitor providers using an `ldapsearch` such as the following:

```
$ bin/ldapsearch --port 1389 bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN cn=monitor \
  --searchScope sub "(objectclass=scim-servlet-monitor-entry)"
```

For example, the following monitor output was produced by a test environment with three distinct SCIM servlet instances, Aleph, Beth, and Gimel. Note that the first instance has a custom resource type called host.

```
$ bin/ldapsearch --baseDN cn=monitor \
  '(objectClass=scim-servlet-monitor-entry)'
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Aleph)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Aleph)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
user-resource-delete-successful: 1
user-resource-put-content-xml: 27
user-resource-query-response-json: 3229836
user-resource-put-403: 5
user-resource-put-content-json: 2
user-resource-get-401: 1
user-resource-put-response-json: 23
user-resource-get-response-json: 5
user-resource-get-response-xml: 7
user-resource-put-400: 2
user-resource-query-401: 1141028
user-resource-post-content-json: 1
user-resource-put-successful: 22
user-resource-post-successful: 1
user-resource-delete-404: 1
user-resource-query-successful: 2088808
```

```
user-resource-get-successful: 10
user-resource-put-response-xml: 6
user-resource-get-404: 1
user-resource-delete-401: 1
user-resource-post-response-json: 1
host-resource-query-successful: 5773268
host-resource-query-response-json: 11576313
host-resource-query-400: 3
host-resource-query-response-xml: 5
host-resource-query-401: 5788152
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Beth)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Beth)
version: 1.2.0
build: 20120105174457Z
revision: 820
serviceproviderconfig-resource-get-successful: 3
serviceproviderconfig-resource-get-response-json: 2
serviceproviderconfig-resource-get-response-xml: 1
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
group-resource-query-successful: 245214
group-resource-query-response-json: 517841
group-resource-query-400: 13711
group-resource-query-401: 258916
user-resource-query-response-json: 107876
user-resource-query-400: 8288
user-resource-get-400: 33
user-resource-get-response-json: 1041
user-resource-get-successful: 2011
user-resource-query-successful: 45650
user-resource-get-response-xml: 1003
user-resource-query-401: 53938
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Gimel)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Gimel)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 1
schema-resource-query-401: 1
schema-resource-query-response-json: 2
user-resource-query-successful: 65
user-resource-get-successful: 4
user-resource-get-response-json: 6
user-resource-query-response-json: 132
user-resource-get-404: 2
```

```
user-resource-query-401: 67
```

About the HTTP Log Publishers

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.
- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.
- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as "'['dd/MM/yyyy:HH:mm:ss Z']'"
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information than the common log format in a format that is familiar to administrators who use the File-Based Access Log Publisher.

The HTTP Detailed Access Log Publisher generates log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-
38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-
d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the `dsconfig` command-line tool help.

Chapter 24

Managing Server SDK Extensions

Topics:

The PingDirectory Server provides support for any custom extensions that you create using the Server SDK. This chapter summarizes the various features and extensions that can be developed using the Server SDK.

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

👉 **Note:** The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Available Types of Extensions

The Server SDK provides support for creating a number of different types of extensions for Ping Identity Server Products, including the PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. Some of those extensions include:

Cross-Product Extensions

Access Loggers
Alert Handlers
Error Loggers
Key Manager Providers
Monitor Providers
Trust Manager Providers
OAuth Token Handlers
Manage Extension Plugins

PingDirectory Server Extensions

Certificate Mappers
Change Subscription Handlers
Extended Operation Handlers
Identity Mappers
Password Generators
Password Storage Schemes
Password Validators
Plugins
Tasks
Virtual Attribute Providers

PingDirectoryProxy Server Extensions

LDAP Health Checks
Placement Algorithms
Proxy Transformations

PingDataSync Server Extensions

JDBC Sync Sources
JDBC Sync Destinations
LDAP Sync Source Plugins
LDAP Sync Destination Plugins
Sync SourcesSync Destinations

- Sync Pipe Plugins

For more information on the Server SDK, see the documentation available in the SDK build.

Chapter 25

Troubleshooting the Server

Topics:

The PingDirectory Server provides a highly-reliable service that satisfies your company's objectives. However, if problems do arise (whether from issues in the Directory Server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

This chapter provides information about how to perform this analysis to help ensure that the problem is resolved as quickly as possible. This chapter presents the following information:

Working with the Collect Support Data Tool

The Directory Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the `collect-support-data` tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool varies between systems. However, the tool always tries to get the same information across all systems for the target Directory Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Server Commands Used in the Collect Support Data Tool

The following presents a summary of the data collectors that the `collect-support-data` tool archives in zip format. If an error occurs during processing, you can re-run the specific data collector command and send the results to your authorized support provider.

Table 66: Directory Server Commands Used in the Collect-Support-Data Tool

Data Collector	Description
status	Runs `status` -F to show the full version information of the Directory Server (Unix, Windows).
server-state	Runs `server-state` to show the current state of the Directory Server process (Unix, Windows).
dsreplication status	Runs `dsreplication status` to show the status of the replicated topology (Unix, Windows). If the --noReplicationStatus option is used, the replication status information is not collected.

JDK Commands Used in the Collect-Support-Data Tool

Table 67: JDK Commands Used in the Collect-Support-Data Tool

Data Collector	Description
jps	Java Virtual Machine Process status tool. Reports information on the JVM (Linux, Windows, Mac OS).
jstack	Java Virtual Machine Stack Trace. Prints the stack traces of threads for the Java process (Linux, Windows, Mac OS).
jstat	Java Virtual Machine Statistics Monitoring Tool. Displays performance statistics for the JVM (Linux, Windows, Mac OS).
jinfo	Displays the Java configuration information for the Java process (Linux, Windows, Mac OS).

Linux Commands Used in the collect-support-data Tool

Table 68: Linux Commands Used in the Collect-Support-Data Tool

Data Collector	Description
tail	Displays the last few lines of a file. Tails the `/var/logs/messages` directory.
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
cat	Concatenates the following files and prints to standard output: /proc/cpuinfo /proc/meminfo /etc/hosts /etc/nsswitch.conf /etc/resolv.conf
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active processed specified by the pid.
top	Prints a list of active processes and how much CPU and memory each process is using.

MacOS Commands Used in the Collect Support Data Tool

Table 69: MacOS Commands Used in the Collect-Support-Data Tool

Data Collector	Description
uname	Prints system, machine, and operating system information.
uptime	Prints the time the server has been up and active.
ps	Prints a snapshot of the current active processes.
system_profiler	Prints system hardware and software configuration.
vm_stat	Prints machine virtual memory statistics.
tail	Displays the last few lines of a file. Tails the `/var/log/system.log` directory.
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
sample	Profiles a process during an interval.

Available Tool Options

The `collect-support-data` tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.
- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.
- **--sequential**. Use this option to diagnose "Out of Memory" errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.
- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, `vmstat`, `iostat`, or `mpstat`). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.
- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, `mpstat`). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.
- **--maxJstacks {number}**. Specifies the number of jstack samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they will impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. The comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain `dsreplication status` information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain `dsreplication status` information.

To Run the Collect Support Data Tool

1. Go to the server root directory.
2. Use the `collect-support-data` tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --serverRoot /opt/PingDirectory --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Directory Server Troubleshooting Information

The Directory Server has a comprehensive default set of log files and monitor entries that are useful when troubleshooting a particular server problem.

Error Log

By default, this log file is available at `logs/errors` below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE msgID=458887
 msg="The Directory Server has started successfully"
```

The category field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL, ADMIN, ADMIN_TOOL, BACKEND, CONFIG, CORE, DSCONFIG, EXTENSIONS, PROTOCOL, SCHEMA, JEB, SYNC, LOG, PLUGIN, PROXY, QUICKSETUP, REPLICATION, RUNTIME_INFORMATION, TASK, THIRD_PARTY, TOOLS, USER_DEFINED, UTIL, VERSION.

The severity field provides information about how severe the server considers the problem to be. Available severities include:

- **DEBUG** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Server provides a separate debug logging facility as described below.
- **INFORMATION** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.
- **MILD_WARNING** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.
- **MILD_ERROR** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.
- **NOTICE** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.
- **SEVERE_WARNING** – Used for problems that the server detects that might lead to bigger problems in the future and should be addressed by administrators.
- **SEVERE_ERROR** – Used for significant problems that have prevented the server from successfully completing processing and are considered important.
- **FATAL_ERROR** – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the NOTICE, SEVERE_WARNING, SEVERE_ERROR, and FATAL_ERROR severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the PROTOCOL category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-
warning,severe-error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the DEBUG severity for a category will not automatically enable logging at the INFORMATION, MILD_WARNING, or MILD_ERROR severities.

The error log configuration may be altered on the fly using tools like `dsconfig`, the Administrative Console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out Log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. The `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug Log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- **enabled** – Indicates whether debug logging is enabled. By default, it is disabled.
- **log-file** – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- **default-debug-level** – Specifies the minimum log level for debug messages that should be written. The default value is “error,” which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include warning, info, and verbose. Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the info debug level automatically enables the warning and error levels.
- **default-debug-category** – Specifies the categories for debug messages that should be written. Some of the most useful categories include caught (provides information and stack traces for any exceptions caught during processing), database-access (provides information about operations performed in the underlying database), protocol (provides information about ASN.1 and LDAP communication performed by the server), and data (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files to help isolate information that might be relevant to a particular problem.

☞ **Note:** Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and then be scoped so that only pertinent debug information is recorded.

Debug targets can be used to further pare down the set of messages generated. For example, you can specify that the debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and interpret the output.

Replication Repair Log

The replication repair log is written to `logs/replication` by default and records information about processing performed by the replication repair service. This log is used to resolve replication conflicts that can arise. For example, if the same entry is modified at the same time on two different systems, or if an attempt is made to create entries with the same DN at the same time on two different systems, the Directory Server records these events.

Config Audit Log and the Configuration Archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the `logs/config-audit.log` file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the `dsconfig` command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.

- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Server maintains complete copies of all previous configurations. These configurations are provided in the `config/archived-configs` directory and are gzip-compressed copies of the `config/config.ldif` file in use before the configuration change was made. The filenames contain time stamps that indicate when that configuration was first used.

Access and Audit Log

The access log provides information about operations processed within the server. The default access log file is written to `logs/access`, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1"
 to="127.0.0.1"
  protocol="LDAP+TLS" clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
  dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
 clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2
  base="ou=People,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL"
  resultCode=0 etime=0.549 entriesReturned=1
[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensi- tive or not useful.

Result messages include a `resultCode` element that indicates whether the operation was successful or if failed and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **origin=replication** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **message** – Text that was included in the `diagnosticMessage` field of the response sent to the client.
- **additionalInfo** – Additional information about the operation that was not included in the response sent back to the client.
- **authDN – DN** of the user that authenticated to the server (typically only included in bind result messages).
- **authzDN – DN** of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **authFailureID** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).
- **authFailureReason** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **responseOID** – OID included in an extended response returned to the client.
- **entriesReturned** – Number of matching entries returned to the client for a search operation.

- **unindexed=true** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The UnboundID LDAP SDK for Java provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify DN) processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to `logs/audit`) in LDIF form.

The PingDirectory Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup Log

The `setup` tool writes a log file providing information about the processing that it performs. By default, this log file is written to `logs/setup.log` although a different name may be used if a file with that name already exists, because the `setup` tool has already been run. The full path to the setup log file is provided when the `setup` tool has completed.

Tool Log

Many of the administrative tools provided with the Directory Server (for example, `import-ldif`, `export-ldif`, `backup`, `restore`, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the `logs/tools` directory (for example, **logs/tools/ import-ldif.log**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

je.info and je.config Files

The primary data store used by the Directory Server is the Oracle Berkeley DB Java Edition (JE). The Directory Server provides two primary sources of information about processing within the database.

The first is logging performed by the JE code itself, and is written into the `je.info.0` file in the server containing the database files (for example, `db/userRoot/je.info.0`). In the event of a problem within JE itself, useful information about the nature of the problem may be written to this log. The level of information written to this log file is controlled by the `db-logging-level` property in the backend configuration object. It uses the standard Java logging framework for logging messages, so the standard SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST levels are available.

The second is configuration information used when opening the database environment. When the backend database environment is opened, then the Directory Server will also write a file named `je.config` in the server containing the database files (for example, `db/userRoot/je.config`) with information about the configuration used.

LDAP SDK Debug Log

This log can be used to help examine the communication between the Directory Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

About the Monitor Entries

While the Directory Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the `cn=monitor` subtree. The types of monitor entries that are available include:

- **General Monitor Entry (cn=monitor)** – Provides a basic set of general information about the server.
- **Active Operations Monitor Entry (cn=Active Operations,cn=monitor)** – Provides information about all operations currently in progress in the server.
- **Backend Monitor Entries (cn={id} Backend,cn=monitor)** – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- **Client Connections Monitor Entry (cn=Client Connections,cn=monitor)** – Provides information about all connections currently established to the server.
- **Connection Handler Monitor Entry (cn={name},cn=monitor)** – Provides information about the configuration of each connection handler and the client connections established to it.
- **Database Environment Monitor Entries (cn={id} Database Environment,cn=monitor)** – Provides statistics and other data from the Oracle Berkeley DB Java Edition database envi- ronment used by the associated backend.
- **Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor)** – Provides infor- mation about the amount of usable disk space available to server components.
- **JVM Memory Usage Monitor Entry (cn=JVM Memory Usage,cn=monitor)** – Provides information about garbage collection activity, the amount of memory available to the server, and the amount of memory consumed by various server components.
- **JVM Stack Trace Monitor Entry (cn=JVM Stack Trace,cn=monitor)** – Provides a stack trace of all threads in the JVM.
- **LDAP Statistics Monitor Entries (cn={name} Statistics,cn=monitor)** – Provides information about the number of each type of operation requested and bytes transferred over the connection handler.
- **Processing Time Histogram Monitor Entry (cn=Processing Time Histogram,cn=monitor)** – Provides information about the number of percent of operations that completed in various response time categories.
- **SSL Context Monitor Entry (cn=SSL Context,cn=monitor)** – Provides information about the available and supported SSL Cipher Suites and Protocols on the server.
- **System Information Monitor Entry (cn=System Information,cn=monitor)** – Provides information about the underlying JVM and system.
- **Version Monitor Entry (cn=Version,cn=monitor)** – Provides information about the Directory Server version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Directory Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Directory Server Troubleshooting Tools

The PingDirectory Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server Version Information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Server software that is in use. If the server is running, then this information can be obtained from the "`cn=Version,cn=monitor`" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.

- Source revision number from which the server was built.
- Build information including build ID with time stamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Jetty, JZlib, SNMP4j (SNMP4J, Agent, Agentx), Groovy, LDAP SDK for Java, and the Server SDK.

LDIF Connection Handler

The Directory Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This Directory Server does not exist by default, but if it is created and an LDIF file is placed in it that contains one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

dbtest Tool

The `dbtest` tool provides a utility that can be used to obtain general information about the data in a backend database. The tool dumps information about entries or keys, and raw data from the database. It can also find keys that have exceeded the entry limit.

For example, the following command can be used to dump a list of all keys in the `objectClass.equality` that have exceeded the entry threshold:

```
$ bin/dbtest dump-database-container \
  --backendID userRoot \
  --baseDN "dc=example,dc=com" \
  --databaseName objectClass.equality \
  --onlyExceedingLimit
```

On a large database, many `dbtest` operations may take a long time to complete, since every record in the associated database is examined. Use the database name option to list a specific database. The following command displays information about the uid.equality database in the `dc=example,dc=com` entry container in the userRoot backend.

```
$ bin/dbtest list-database-containers -n userRoot -b "dc=example,dc=com" -d
 uid.equality
```

Index Key Entry Limit

Indexes have keys that maintain a list of matching entries, up to the index entry limit. When that limit is reached, the key will not contain or maintain that list, and will just maintain a count of matching entries. To determine if index keys are approaching their limit, use either the `dbtest` tool or the `verify-index` tool.

While the `dbtest` tool can be used to gather general imformation, the `verify-index` tool provides statistical data about the percent of entries covered by the keys.

For example, the following command can be used to retrieve a list of keys that have exceeded the entry threshold:

```
$ bin/verify-index \
  --baseDN dc=example,dc=com \
  --listKeysExceedingIndexEntryLimit
```

The following is a sample of the data returned:

```
 [12:06:05]  Checked 6003 entries and found 0 error(s) in 2 seconds (average
 rate 2453.2/sec)
```

```
[12:06:05]  Statistics for records that have exceeded the entry limit:
[12:06:05]  The st.equality index has 48 such record(s) limit=100 min=103
max=152 median=118
[12:06:05]  1. or (152 entries / 2.53% of all entries)
[12:06:05]  2. ma (132 entries / 2.20% of all entries)
 .....
[12:06:05]  The id2subtree index has 2 such record(s) limit=4000 min=6000
max=6002 median=6001
[12:06:05]  1. 1 => dc=example,dc=com (6002 entries / 99.98% of all entries)
 .....
[12:06:05]  The id2children index has 1 such record(s) limit=4000 min=6000
max=6000 median=6000
[12:06:05]  1. 2 => ou=People,dc=example,dc=com (6000 entries / 99.95% of all
entries)
[12:06:05]  The objectClass.equality index has 4 such record(s) limit=4000
min=6001 max=6003 median=6001
[12:06:05]  1. top (6003 entries / 100.00% of all entries)
 .....
```

Embedded Profiler

If the Directory Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the `dsconfig` tool or the Administrative Console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The `profile-action` property for this configuration object can have one of the following values:

- **start** – Indicates that the embedded profiler should start capturing data in the background.
- **stop** – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a `logs/profile{timestamp}` file.
- **cancel** – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the `profiler-viewer` tool. This tool can operate in either a text-based mode, in which case it dumps a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

To Invoke the Profile Viewer in Text-based Mode

- Run the `profile-viewer` command and specify the captured log file using the --fileName option.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

To Invoke the Profile Viewer in GUI Mode

- Run the `profile-viewer` command and specify the captured log file using the --fileName option. To invoke GUI mode, add the option --useGUI.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Oracle Berkeley DB Java Edition Utilities

The Oracle Berkeley DB Java Edition (JE) itself provides a number of utilities that can be used for performing various types of low-level debugging in the database environment. These utilities should generally not be used unless you are advised to do so by your authorized support provider, but they provide access to information about the underlying database environment that is not available through any other means.

Troubleshooting Resources for Java Applications

Because the PingDirectory Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Server. In many cases, obtaining the full benefit of these tools requires access to the Directory Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java Troubleshooting Tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the PingDirectory Server.

jps

The `jps` tool is a Java-specific version of the UNIX `ps` tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- **-v** – Includes the arguments passed to the JVM for the processes that are listed.
- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

jstack

The `jstack` tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can have user-friendly names (as do the threads used by the PingDirectory Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where {processID} is the process ID of the target JVM as returned by the `jps` command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the java command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The "-l" option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The “-m” option can be used to include native frames in the stack trace.

jmap

The jmap tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native pmap tool provided by many operating systems. As with the jstack tool, jmap can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the jhat tool
- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.
- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the ":live" portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

jhat

The jhat (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory Server encounters an out of memory error (as a result of the "-XX:+HeapDumpOnOutOfMemoryError" JVM option) or from the use of the jmap command with the "-dump" option.

The jhat tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The jhat tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the jhat web server to begin listening on port 7000. It can be accessed in a browser at http://localhost:7000 (or http://address:7000 from a remote system). An alternate port number can be specified using the "-port" option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL http://localhost:7000/oql/ (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at http://localhost:7000/oqlhelp/.

jstat

The jstat tool is used to obtain a variety of statistical information from the JVM, much like the vmstat utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The {interval} option specifies the length of time in milliseconds between lines of output. The {processID} option specifies the process ID of the JVM used to run the Directory Server, which can be obtained by running jps as mentioned previously. The {type} option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.
- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.

- **-gccapacity** – Provides information about memory region capacities.

Java Diagnostic Information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

JVM Crash Diagnostic Information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the base directory of the Directory Server installation. This file includes information on the underlying cause of the JVM crash, information about the threads running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system.

Troubleshooting Resources in the Operating System

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the Directory Server, and in others system, tools can help narrow down the cause of the problem.

Identifying Problems with the Underlying System

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. To look for problems in the underlying system view the system log file (`/var/log/messages` on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.

Examining CPU Utilization

Observing CPU utilization for the Directory Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU Utilization

To investigate CPU consumption of the system as a whole, use the `vmstat` command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU Utilization

To investigate CPU consumption on a per-CPU basis, use the `mpstat` command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "-P ALL" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but mpstat reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-Process Utilization

To investigate CPU consumption on a per-process basis, use a command such as the top utility on Linux. If a process other than the Java process used to run the Directory Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Server to run effectively.

Examining Disk Utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The iostat tool may be used to obtain information about the disk activity on the system.

On Linux systems, iostat should be invoked with the "-x" argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but to obtain an initial feel for how busy the underlying disks are, look at the "%util" column on Linux. This field shows the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a tmpfs filesystem).

Examining Process Details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard ps tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard ps command displays only a limited set of the arguments based on available space in the terminal window.

pstack

The pstack command can be used to obtain a native stack trace of all threads in a process. While a native stack trace might not be as user-friendly as a Java stack trace obtained using jstack, it includes threads that are not available in a Java stack trace. For example, the command displays those threads used to perform garbage collection and other housekeeping tasks. The general usage for the pstack command is:

```
pstack {processID}
```

dbx / gdb

A process debugger provides the ability to examine a process in detail. Like `pstack`, a debugger can obtain a stack trace for all threads in the process, but it also provides the ability to examine a process (or core file) in much greater detail, including observing the contents of memory at a specified address and the values of CPU registers in different frames of execution. The GNU debugger `gdb` is widely-used on Linux systems.

Note that using a debugger against a live process interrupts that process and suspends its execution until it detaches from the process. In addition, when running against a live process, a debugger has the ability to actually alter the contents of the memory associated with that process, which can have adverse effects. As a result, it is recommended that the use of a process debugger be restricted to core files and only used to examine live processes under the direction of your authorized support provider.

pfiles / lsof

To examine the set of files that a process is using (including special types of files, like sockets), you can use a tool such as `lsof` on Linux systems, (

```
lsof -p {processID}
```

)

Tracing Process Execution

Content-Reference to:

If a process is unresponsive but is consuming a nontrivial amount of CPU time, or if a process is consuming significantly more CPU time than is expected, it might be useful to examine the activity of that process in more detail than can be obtained using a point-in-time snapshot. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

This can be accomplished using the `strace` tool on Linux (

```
strace -f -p {processID}
```

).

Consult the `strace` manual page for additional information.

Problems with SSL Communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-server` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Examining Network Communication

Content-Reference to:

Because the PingDirectory Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Server itself can provide details about its interaction with clients by enabling debugging for the protocol or data debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the *tcpdump* tool on Linux, can be used to accomplish this.

There are many options that can be used with these tools, and their corresponding manual pages will provide a more thorough explanation of their use. However, to perform basic tracing to show the full details of the packets received for communication on port 389 with remote host 1.2.3.4, the following command can be used on Linux:

```
tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

It does not appear that the `tcpdump` tool provides support for LDAP parsing. However, it is possible to write capture data to a file rather than displaying information on the terminal (using "-w {path}" with `tcpdump`), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access.

Common Problems and Potential Solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General Methodology to Troubleshoot a Problem

When a problem is detected, Ping Identity recommends using the following general methodology to isolate the problem:

1. Run the `bin/status` tool or look at the server status in the Administrative Console. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts.
2. Look in the server logs. In particular, view the following logs:

   logs/errors
   logs/failed-ops
   logs/expensive-ops
3. Use system commands, such as `vmstat` and `iostat` to determine if the server is bottle-necked on a system resource like CPU or disk throughput.
4. For performance problem (especially intermittent ones like spikes in response time), enabling the `periodic-stats-logger` can help to isolate problems, because it stores important server performance information on a per-second basis. The `periodic-stats-logger` can save the information in a csv-formatted file that can be loaded into a spreadsheet. The information this logger makes available is very configurable. You can create multiple loggers for different types of information or a different frequency of logging (for example, hourly data in addition to per-second data). For more information, see "Profiling Server Performance Using the Periodic Stats Logger".
5. For replication problem, run `dsreplication status` and look at the `logs/replication` file.
6. For more advanced users, run the `collect-support-data` tool on the system, unzip the archive somewhere, and look through the collected information. This is often useful when administrators most familiar with the PingData Platform do not have direct access to the systems where the production servers are running. They can examine the `collect-support-data` archive on a different server. For more information, see Using the Collect Support Data Tool.

☞ **Important:** Run the `collect-support-data` tool whenever there is a problem whose cause is not easily identified, so that this information can be passed back to your authorized support provider before corrective action can be taken.

The Server Will Not Run Setup

If the `setup` tool does not run properly, some of the most common reasons include the following:

A Suitable Java Environment Is Not Available

The PingDirectory Server requires that Java be installed on the system and made available to the server, and it must be installed prior to running `setup`. If the `setup` tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the `setup` tool should be invoked with an explicitly-defined value for the *JAVA_HOME* environment variable that specifies the path to the Java installation that should be used. For example:

```
env JAVA_HOME=/ds/java ./setup
```

If this still does not work for some reason, then it can be that the value specified in the provided *JAVA_HOME* environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Oracle Berkeley DB Java Edition Is Not Available

If the version of the Directory Server that you are using was not provided with the Oracle Berkeley DB Java Edition library, then it must be manually downloaded and the appropriate JAR file placed in the `lib` directory before running setup. See the `lib/downloading-je.txt` file for instructions on obtaining the appropriate library.

Unexpected Arguments Provided to the JVM

If the `setup` script attempts to launch the java command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running setup, but this might not be the case if either the *JAVA_ARGS* or *UNBOUNDID_JAVA_ARGS* environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run `setup`:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Used

The `setup` tool is only intended to provide the initial configuration for the Directory Server. It refuses to run if it detects that the `setup` tool has already been run, or if an attempt has been made to start the Directory Server prior to running the `setup` tool. This protects an existing Directory Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run `setup` in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run `setup` failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the `setup` program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The Server Will Not Start

If the Directory Server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running

Only a single instance of the Directory Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other administrative operations can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Server could not acquire an exclusive lock on file
/ds/PingDirectory/locks/server.lock: The exclusive lock requested for file
/ds/PingDirectory/locks/ server.lock was not granted, which indicates
that another process already holds a shared or exclusive lock on that
file. This generally means that another instance of this server is already
running
```

If the Directory Server is not running (and is not in the process of starting up or shutting down) and there are no other tools running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the `locks` directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the filesystem.

There Is Not Enough Memory Available

When the Directory Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Server might need to be re-configured to use a smaller amount of memory than had been previously configured.
- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Server, or reconfigure the Directory Server to reduce the amount of memory that it attempts to use.
- The Directory Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the `vmstat` command and wait until the amount of free memory stops growing before attempting to restart the server.
- If the system is configured with one or more memory-backed filesystems, then look to see if there are any large files that can be consuming a significant amount of memory in any of those locations. If so, then remove them or relocate them to a disk-based filesystem.
- For Linux systems only, if there is a mismatch between the huge pages setting for the JVM and the huge pages reserved in the operating system.

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An Invalid Java Environment or JVM Option Was Used

If an attempt to start the Directory Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the `config/java.properties` file to reference to path to the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the

new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.

- If an *UNBOUNDID_JAVA_HOME* or *UNBOUNDID_JAVA_BIN* environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the `config/java.properties` file. Similarly, if an *UNBOUNDID_JAVA_ARGS* environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the *UNBOUNDID_JAVA_HOME*, *UNBOUNDID_JAVA_BIN*, and *UNBOUNDID_JAVA_ARGS* environment variables before trying to start the server.

Note that any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, then it can be necessary to remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option Was Provided

There are a small number of arguments that are provided when running the `bin/start-server` command, but in most cases, none are required. If one or more command-line arguments were provided for the `bin/start-server` command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration

If a change is made to the Directory Server configuration using an officially-supported tool like `dsconfig` or the Administrative Console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the `logs/config-audit.log` file to see what recent configuration changes have been made with the server online, or in the `config/archived-configs` directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the --useLastKnownGoodConfig option:

```
$ bin/start-server --useLastKnownGoodConfig
```

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the `config/config.ldif` file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.

Note: In addition to manually-editing the config file, you can look at previous achived configurations to see if the most recent one works. You can also use the `ldif-diff` tool to compare the configurations in the archive to the current configuration to see what is different.

You Do Not Have Sufficient Permissions

The Directory Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/PingDirectory
```

The Server Has Crashed or Shut Itself Down

You can first check the current server state by using the `bin/server-state` command. If the Directory Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using “kill -9”), then messages are written to the `error` and `server.out` logs explaining the reason for the shutdown.
- The Directory Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the `uptime` command on the underlying system shows that it was recently booted.
- The Directory Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the `error` and `server.out` logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Server was running crashed. If this happens, then the JVM should dump a fatal error log (a `hs_err_pid{processID}.log` file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

Conditions for Automatic Server Shutdown

All PingData servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The Directory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
           --set unrecoverable-database-error-mode:initiate-server-shutdown
```

The Server Will Not Accept Client Connections

You can first check the current server state by using the `bin/server-state` command. If the Directory Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Server is not running.
- The underlying system on which the Directory Server is installed is not running.
- The Directory Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.

- If the Directory Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.
- If the Directory Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The Server is Unresponsive

You can first check the current server state by using the `bin/server-state` command. If the Directory Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the `jstack` command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

  A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

  Note: If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.
- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the `jstack` command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.
- If the JVM in which the Directory Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a `pstack` of the native process should show that most threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.
- If the JVM in which the Directory Server is running has hung for some reason, then the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like `snoop` or `tcpdump` should show that packets sent to the system on which the Directory Server is running are not receiving TCP acknowledgement.
- If the system on which the Directory Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is Slow to Respond to Client Requests

If the Directory Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see "Profiling Server Performance Using the Periodic Stats Logger". The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second. You can also run the `bin/status` command or view the status in the Administrative Console to see the Directory Server's Work Queue information (also see the next bullet point).
- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the `bin/status` command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

  ```
             --- Work Queue ---
             : Recent : Average : Maximum
  -----------:--------:---------:--------
  Queue Size : 10  : 1       : 10
  % Busy     : 17     : 14      : 100
  ```

  You can also view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second.
- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, `iostat` reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see Disk-Bound Deployments.
- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.
- If the Directory Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, `iostat` should report very high utilization for disks used to hold swap space, and commands like `cat /proc/meminfo` on Linux can report a large amount of swap memory in use. Another cause of swapping is if swappiness is not set to 0 on Linux. For more information, see Disable File System Swapping.
- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server Returns Error Responses to Client Requests

If a large number of client requests are receiving error responses, then view the `logs/failed-ops` log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.
- If a portion of the Directory Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxies or load balancers or both to route requests away from the affected server. As of Directory Server version 3.1 or later, the Directory Server will indicate that it is in a degraded status and the Directory Proxy Server will route around it.
- If the Directory Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.
- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

The Server Must Disconnect a Client Connection

If a client connection must be disconnected due to the expense of the client's request, such as an unindexed search across a very large database, perform the following:

- Find the client's connection ID by looking in the `cn=Active Operations,cn=monitor monitor` entry.

```
$ bin/ldapsearch -baseDN cn=monitor "cn=active operations" \
  --bindDN "cn=directory manager"  \
  --bindPassword password
```

- The monitor entry will contain attribute values for `operation-in-progress`, which look like an access log message. Look for the value of `conn` in the client request that should be disconnected. In the following example, the client to be disconnected is requesting a search for `(description=expensive)`, which is on connection 6.

```
dn: cn=Active Operations,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-active-operations-monitor-entry
objectClass: extensibleObject
cn: Active Operations
num-operations-in-progress: 2
operation-in-progress: [15/Dec/2014:10:55:35 -0600] SEARCH conn=6 op=3
 msgID=4
     clientIP="10.8.4.21" authDN="cn=app1,ou=applications,dc=example,dc=com"
 base="dc
     =example,dc=com" scope=wholeSubtree filter="(description=expensive)"
 attrs="A
     LL" unindexed=true
operation-in-progress: [15/Dec/2014:10:56:11 -0600] SEARCH conn=7 op=1
 msgID=2
     clientIP="127.0.0.1" authDN="cn=Directory Manager,cn=Root
 DNs,cn=config" base="c
     n=monitor" scope=wholeSubtree filter="(cn=active operations)"
 attrs="ALL"
     num-persistent-searches-in-progress: 0
```

- With the connection ID value, create a file with the following contents, named `disconnect6.ldif`.

```
dn: ds-task-id=disconnect6,cn=scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-disconnect
ds-task-disconnect-connection-id: 6
ds-task-id: disconnect6
ds-task-class-name:
 com.unboundid.directory.server.tasks.DisconnectClientTask
```

- This LDIF file represents a task entry. The connection ID value 6 is assigned to `ds-task-disconnect-connection-id`. The value for `ds-task-id` value does not follow a specific convention. It must be unique among other task entries currently cached by the server.
- Disconnect the client and cancel the associated operation by adding the task entry to the server:

```
$ bin/ldapmodify --filename disconnect6.ldif  \
  --defaultAdd --bindDN "cn=directory manager" \
  --bindPassword password
```

The Server is experiencing problems with replication

If replication does not appear to be functioning properly, then first check the `dsreplication status` command, which shows all of the servers that are replicating and whether they are back-logged or not. Next, you can check the server error log, replication repair log, and replication monitor entries may provide information about the nature of the underlying problem. Potential reasons that replication may not be functioning as expected include the following:

- Replication has not yet been configured between systems or has been disabled.
- If a server has been offline for a period of time or has fallen far enough behind such that it is missing changes, which are no longer present in any of the replication databases, then that server must be re-initialized with an up-to-date copy of the data from another server.
- If the environment is comprised of a heterogeneous set of systems, then it is possible that some of the systems might not be able to keep up with the maximum throughput achieved by other servers in the topology. In such cases, the slower servers might not be fast enough to remain in sync with the other servers in the environment.
- If the environment contains systems in multiple data centers and the network links between the data centers are insufficient for the volume of changes that must be processed, then servers might not be able to remain in sync under a high volume of changes.
- A network or firewall configuration problem has arisen, which prevents or interferes with communication between servers.
- An internal problem within the server has caused replication to stop functioning properly. The Directory Server logs the event in the error log in this case. Run the `collect-support-data` tool, so that the details of the problems can be passed to your authorized support provider. Then, try restarting the Directory Server.

How to Regenerate the Server ads-certificate

At setup time, the server generates a private key and certificate for use when secure communication between servers is required. This certificate, `ads-certificate`, is stored in `config/ads-truststore` and should typically remain unchanged for the life of the server deployment. If the need arises for a new ads-certificate to be created, say because the server-root has been copied to a new host, then the private key and certificate will be recreated by the startup process if the `config/ads-truststore` and `config/ads-truststore.pin` files are first manually removed while the server is offline. Note that if replication is enabled, the server must have replication disabled before regeneration of the ads-certificate.

For example, the Directory Server allows easy copying of its installation, which can then be used to install another server instance. If a server (ldap1.example.com:389) is enabled with its own copy (ldap2.example.com:389), `dsreplication` will exit with the following error message:

```
Replication cannot be enabled between servers ldap1.example.com:389 and
 ldap2.example.com:389
because they are using the same instance key.
```

The solution is to stop the server, remove `config/adstruststore` and `config/adstruststore.pin` and re-start the server. Upon startup, a new `adstruststore`, containing the server's instance key, will be generated. Then, you can re-run `dsreplication enable` to set up replication between the two servers.

The Server behaves differently from Sun/Oracle

After migrating from a Sun/Oracle configuration to a PingDirectory Server, follow the tuning procedures in Sun/ Oracle Compatibility if the Directory Server behaves differently from the Sun/Oracle server.

Troubleshooting ACI Evaluation

The Directory Server provides the ability to collect debug information related to ACI evaluation for any operation by enabling the Debug ACI Logger. The Debug ACI Logger is highly configurable and can be scoped to trace very specific request operations in order to narrow on any ACI issue that may arise in the field. Parameters for non-request operations, such as `log-connects`, `log-disconnects`, `log-security-negotiation`, `log-results`, `log-assuance-completed`, `log-search-entries`, `log-search-references`, `log-intermediate-responses` are set to `false` by default and should remain so.

Here is an example to enable the Debug ACI Logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Debug ACI Logger" \
  --set enabled:true
```

Using this debug tracer is often more efficient by limiting the output using request and result criteria to match specific types of operations. An example result criteria for operations that fail due to insufficient access rights can be added to the logger as follows:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Debug ACI Logger" \
  --set "result-criteria:Insufficient Access Rights"
```

Once the logger is enabled, all matching operations begin writing ACI evaluation traces to the log file. The amount of information is quite large for each evaluation that is done. However, this information is useful if there is an ACI issue that is difficult to resolve. Most operations result in multiple "ACI DEBUG" traces in the log, since it usually requires multiple ACI rights to perform an operation, each of which requires a separate evaluation. In particular, you can expect a lot of debug tracing when dealing with ACIs for controls, extended operations, and proxied authorization.

The ACI DEBUG traces contain the following pieces of information:

- **Operation**. Specifies a dump of the operation object that you can use to correlate to the original request operation.
- **ACI Container**. Specifies the context of the ACI evaluation being performed.

  Client Entry. Specifies an LDIF dump of the client request access.
  Resource Entry. Specifies an LDIF dump of the target resource.
  isProxiedAuth. Specifies if the client is attempting to proxy as another user.
  Original Auth. Specifies the original client DN if authorization is currently via the proxy.
  Rights. Specifies a list of the ACI rights being requested on the resource entry.
  Control. Specifies the OIDs when evaluating ACIs for a control.
  ExtOp. Specifies the OIDs when evaluating ACIs for an extended operation.
- **ACI Canidates**. Specifies a list of all the ACIs known to this operation, sorted by origin.
- **Applicable ACIs**. Specifies a list of ACIs relevant to the current evaluation. These ACIs are separated by type into "Denies" and "Allows".
- **Deny ACI Evaluations**. Specifies the results of evaluating each "deny" ACI. If any of these evaluate to TRUE, then the operation will be denied.
- **Allow ACI Evaluations**. Specifies the results of evaluating each "allow" ACI. At least one of these must evaluate to "TRUE" or the operation will be denied.

For users with the `bypass-acl` privilege, the Debug ACI Logger will not provide any ACI debug tracing since evaluations are not done for those operations. However, you will see the following trace if you have ACI debugging enabled (`debug-aci-enabled` is set to TRUE) for those operations:

```
Bypassing ACL Evaluation for Operation
```

To avoid unnecessary tracing of these operations, the "Debug ACI Logger" uses a "Client Connection Criteria" called "Clients subject to Access Control" that excludes requests from users with the `bypass-acl` privilege. It is recommended that you create and use your own criteria which specifically targets the clients that you are trying to debug in order to make analyzing the tracing output easier.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Restricted Clients" \
  --type simple \
  --set none-included-user-privilege:bypass-acl
```

Problems with the Administrative Console

If a problem arises when trying to use the Administrative Console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with the Directory Proxy Server instance using the Administrative Console, then the access and error logs for that Directory Server instance might provide additional information about the underlying problem.

Problems with the Administrative Console: JVM Memory Issues

Console runs out of memory (PermGen). An inadequate PermSize setting in the server, while hosting web applications like the Administrative Console may result in errors like this in the error log:

```
[02/Mar/2016:07:50:27.017 -0600] threadID=2 category=UTIL
    severity=SEVERE_ERROR msgID=-1 msg="The server experienced an unexpected
    error. Please report this problem and include this log file.
    OutOfMemoryError: PermGen space
    ()\ncom.unboundid.directory.server.core.DirectoryServer.uncaughtException
    (DirectoryServer.java:15783)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1057)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.Thread.dispatchUncaughtException
    (Thread.java:1986)\nBuild revision: 22496\n"
```

This is only relevant for servers running Java 7.

Problems with the HTTP Connection Handler

When problems with the HTTP Connection Handler occur, first look at the HTTP connection handler log to diagnose the issue. The following section shows HTTP log examples when various errors occur.

- **Failed Request Due to a Non-Existent Resource**. The server receives a status code 404, which indicates the server could not match the URI.

  ```
  [15/Mar/2012:17:39:39 -0500] RESULT requestID=0 from="10.2.1.113:52958"
  method="GET" url="https://10.2.1.113:443/Aleph/Users/uid=user.1,ou=people,
  dc=example,dc=com" requestHeader="Host: x2270-11.example.lab"
  requestHeader="Accept: */*" requestHeader="User-Agent: curl/7.21.6
  (i386-pc-centos2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22
  libssh2/1.2.7" authorizationType="Basic" statusCode=404 etime=81.484
  ```

```
responseContentLength=103 responseHeader="Access-Control-Allow-
Credentials:true"
responseContentType="application/json"
```

- **Failed Request due to a Malformed Request Body**. The server receives a status code 400, which indicates that the request had a malformed syntax in its request body.

```
[15/Mar/2012:17:47:23-0500] RESULT requestID=10 from="10.2.1.113:55284"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
x2270-11.example.lab" requestHeader="Expect: 100-continue"
requestHeader="Accept: */*" requestHeader="Content-Type: application/json"
requestHeader="User-Agent: curl/ 7.21.6 (i386-pc-centos2.10) libcurl/7.21.6
OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
 authorizationType="Basic"
requestContentType="application/json" requestContentLength=5564
 statusCode=400
etime=15.272 responseContentLength=133 responseContentType="application/
json"
```

- **Failed Request due to an unsupported HTTP method**. The server receives a status code 405, which indicates that the specified method (e.g., "PATCH") in the request line is not allowed for the resource identified in the URI.

```
[15/Mar/2012:17:48:59-0500] RESULT requestID=11 from="10.2.1.113:55763"
method="PATCH" url="https://10.2.1.113:443/Aleph/Users"
 requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="Content-
Type:
application/json" requestHeader="User-Agent: curl/7.21.6 (i386-pc-
centos2.10)
libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
authorization-Type="Basic" requestContentType="application/json"
 statusCode=405
etime=6.807 responseContentLength=0 responseHeader="Allow: POST, GET,
 OPTIONS, HEAD"
```

- **Failed Request due to an Unsupported Media Type**. The server receives a status code 415, which indicates that the request entity is in a format that is not supported by the requested resource.

```
[15/Mar/2012:17:44:45-0500] RESULT requestID=4 from="10.2.1.113:54493"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="Content-
Type:
application/atom+xml" requestHeader="User-Agent: curl/7.21.6 (i386-pc-
centos2.10)
libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
authorizationType="Basic" requestContentType="application/atom+xml"
requestContentLength=3 statusCode=415 etime=6.222
 responseContentLength=1402
responseHeader="Cache-Control: must-revalidate,no-cache,no-store"
responseContentType="text/html;charset=ISO-8859-1"
```

- **Failed Request due to an Authentication Error**. The server receives a status code 401, which indicates that the request requires user authentication.

```
[15/Mar/2012:17:46:06-0500] RESULT requestID=8 from="10.2.1.113:54899"
method="GET" url="https://10.2.1.113:443/Aleph/Schemas"
 requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="User-
Agent:
curl/7.21.6 (i386-pc-centos2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5
libidn/1.22 libssh2/ 1.2.7" authorizationType="Basic" statusCode=401
etime=2.751 responseContentLength=63 responseHeader="WWW-Authenticate:
 Basic
realm=SCIM" responseHeader="Access-Control-Allow-Credentials: true"
responseContentType="application/json"
```

Virtual Process Size on RHEL6 Linux is Much Larger than the Heap

Red Hat Linux introduced a change in glib 2.11 that creates larger per-thread address spaces aligned at 64MB. This change results in a virtual process size much larger than those seen in previous versions of `glibc`. This is not considered a bug by RedHat, as noted in *https://bugzilla.redhat.com/show_bug.cgi?id=640286*, and does not affect the physical memory needed by the server process. To see the version of `glibc` on your system, use the command `yum info glibc`.

Providing Information for Support Cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the `collect-support-data` tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

Chapter

26

Command-Line Tools

Topics:

The PingDirectory Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

This chapter presents the following topics:

Using the Help Option

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. You can view detailed argument options and examples by typing --help with the command.

```
bin/dsconfig --help
```

For those utilities that support additional subcommands (for example, `dsconfig`), you can get a list of the subcommands by typing --help-subcommands.

```
bin/dsconfig --help-subcommands
```

You can also get more detailed subcommand information by typing --help with the specific subcommand.

```
bin/dsconfig list-log-publishers --help
```

Note: For detailed information and examples of the command-line tools, see the *Ping Identity Directory Server Command-Line Tool Reference*.

Available Command-Line Utilities

The Directory Server provides the following command-line utilities, which can be run directly in interactive, non-interactive, or script mode.

Table 70: Command-Line Utilities

Command-Line Tools	Description
audit-data-security	Performs an internal task that examines all or a subset of entries in the server, writing a series of reports on potential risks with the data. Reports are written to the output directory organized by backend name and audit items.
authrate	Perform repeated authentications against the Directory Server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more directory server backends. This utility also supports the use of a properties file to pass predefined command-line arguments. See Managing the tools.properties File for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a ZIP archive that can be sent to a technical support representative.
config-diff	Generate a summary of the configuration changes in a local or remote server instance. The tool can be used to compare configuration settings when troubleshooting issues, or when verifying configuration settings on new servers.
create-rc-script	Create a Run Control (RC) script that may be used to start, stop, and restart the Directory Server on UNIX-based systems.
create-recurring-task	Create a recurring task to run on the server. Tasks can be created for backups, LDIF exports, a statically defined task, or a third-party task.
create-recurring-task-chain	Create a chain of recurring tasks to run on the server.

Command-Line Tools	Description
dbtest	Inspect the contents of Directory Server backends that store their information in Oracle® Berkeley DB Java Edition databases.
deliver-one-time-password	Submit a "deliver one-time password" extended request, OID 1.3.6.1.4.1.30221.2.6.24, to the server which results in a the generation of a one-time password which is delivered out-of-band to the specified user. This tool can be used to test the UNBOUNDID-DELIVERED-OTP SASL mechanism.
dsconfig	View and edit the Directory Server configuration.
dsjavaproperties	Configure the JVM arguments used to run the Directory Server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and *JAVA_HOME* environment variable.
dsreplication	Manage data replication between two or more Directory Server instances.
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Directory Server.
encode-password	Encode user passwords with a specified storage scheme or determine whether a given clear-text value matches a provided encoded password.
encryption-settings	Manage the server encryption settings database.
enter-lockdown-mode	Request that the Directory Server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege.
export-ldif	Export data from the Directory Server backend in LDIF form.
identify-references-to-missing-entries	Identify entries containing one or more attributes that reference entries that do not exist. This may require the ability to perform unindexed searches and/or the ability to use the simple paged results control.
identify-unique-attribute-conflicts	Identify unique attribute conflicts. The tool may identify values of one or more attributes that are supposed to exist only in a single entry but are found in multiple entries.
import-ldif	Import LDIF data into the Directory Server backend.
ldap-diff	Compare the contents of two LDAP servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform LDAP compare operations in the Directory Server.
ldapdelete	Perform LDAP delete operations in the Directory Server.
ldapmodify	Perform LDAP modify, add, delete, and modify DN operations in the Directory Server.
ldappasswordmodify	Perform LDAP password modify operations in the Directory Server.
ldapsearch	Perform LDAP search operations in the Directory Server.
ldif-diff	Compare the contents of two LDIF files, the output being an LDIF file needed to bring the source file in sync with the target.
ldifmodify	Apply a set of modify, add, and delete operations against data in an LDIF file.
ldifsearch	Perform search operations against data in an LDIF file.
leave-lockdown-mode	Request that the Directory Server leave lockdown mode and resume normal operation.

Command-Line Tools	Description
list-backends	List the backends and base DNs configured in the Directory Server.
make-ldif	Generate LDIF data based on a definition in a template file.
manage-account	Access and alter password policy state properties for user entries.
manage-extension	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the PingDirectory Server. Extension bundles are installed from a zip archive or file system directory. PingDirectory Server will be restarted if running to activate the extension(s).
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Directory Server.
migrate-ldap-schema	Migrate schema information from an existing LDAP server into a PingDirectory Server instance.
migrate-sun-ds-config	Update an instance of the PingDirectory Server to match the configuration of an existing Sun Java System 5.x, 6.x, or 7.x directory instance.
modrate	Perform repeated modifications against a Directory Server.
move-subtree	Move a subtree entries or a single entry from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
profile-viewer	View information in data files captured by the Directory Server profiler.
re-encode-entries	Initiate a task that causes a local DB backend to re-encode all or a specified subset of the entries that it contains. The tool does not alter the entries themselves but provides a useful mechanism for applying significant changes to the way that entries are stored in the backend (e.g., to apply encoding changes if a feature like data encryption or uncached attributes or entries is enabled).
rebuild-index	Rebuild index data within a backend based on the Berkeley DB Java Edition. Note that this tool uses different approaches to rebuilding indexes based on whether it is running in online mode (as a task) rather than in offline mode. Running in offline mode will often provide significantly better performance. Also note that running in online mode will prevent the server from using that index while the rebuild is in progress, so some searches may behave differently while a rebuild is active than when it is not.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Directory Server backend.
restore	Restore a backup of the Directory Server backend.
revert-update	Returns a server to the version before the last update was performed.
review-license	Review and/or indicate your acceptance of the product license.
scramble-ldif	Obscure the contents of a specified set of attributes in an LDIF file.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
searchrate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Directory Server process.
setup	Perform the initial setup for the Directory Server instance.
start-server	Start the Directory Server.

Command-Line Tools	Description
status	Display basic server information.
stop-server	Stop or restart the Directory Server.
subtree-accessibility	List or update the a set of subtree accessibility restrictions defined in the Directory Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
uninstall	Uninstall the Directory Server.
update	Update the Directory Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the `update` tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update is performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the `revert-update` tool for information on reverting an update.
validate-acis	Validates a set of access control definitions contained in an LDAP server (including Sun/Oracle DSEE instances) or an LDIF file to determine whether they are acceptable for use in the Directory Server. Note that the output generated by this tool will be in LDIF format, but each entry in the output will have exactly one ACI, so entries that have more than one ACI will appear multiple times in the output with different ACI values.
validate-file-signature	Validate the signature information in a signed text file, such as a signed log file or a signed LDIF export.
validate-ldif	Validate the contents of an LDIF file against the server schema.
verify-index	Verify that indexes in a backend using the Oracle Berkeley DB Java Edition are consistent with the entry data contained in the database.

Managing the tools.properties File

The PingDirectory Server supports the use of a tools properties file that simplifies command-line invocations by reading in a set of arguments for each tool from a text file. Each property is in the form of name/value pairs that define predetermined values for a tool’s arguments. Properties files are convenient when quickly testing the Directory Server in multiple environments.

The Directory Server supports two types of properties file: default properties files that can be applied to all command-line utilities or tool-specific properties file that can be specified using the --propertiesFilePath option. You can override all of the Directory Server's command-line utilities with a properties file using the `config/tools.properties` file.

Creating a Tools Properties File

You can create a properties file with a text editor by specifying each argument, or option, using standard Java properties file format (name=value). For example, you can create a simple properties file that define a set of LDAP connection parameters as follows:

```
hostname=server1.example.com
port=1389
```

```
bindDN=cn=Directory\ Manager
bindPassword=secret
baseDN=dc=example,dc=com
```

Next, you can specify the location of the file using the --propertiesFilePath /path/to/ File option with the command-line tool. For example, if you save the previous properties file as `bin/mytool.properties`, you can specify the path to the properties file with `ldapsearch` as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/mytools.properties "(objectclass=*)"
```

Properties files do not allow quotation marks of any kind around values. Any spaces or special characters should be escaped. For example,

```
bindDN=cn=QA\ Managers,ou=groups,dc=example,dc=com
```

The following is not allowed as it contains quotation marks:

```
bindDN=cn="QA Managers,ou=groups,dc=example,dc=com"
```

Tool-Specific Properties

The Directory Server also supports properties for specific tool options using the format: `tool.option=value`. Tool-specific options have precedence over general options. For example, the following properties file uses `ldapsearch.port=2389` for `ldapsearch` requests by the client. All other tools that use the properties file uses `port=1389`.

```
hostname=server1.example.com
port=1389
ldapsearch.port=2389
bindDN=cn=Directory\ Manager
```

Another example using the `dsconfig` configuration tool is as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
dsconfig.bindPasswordFile=/ds/config/password
```

Note: The `.bindPasswordFile` property requires an absolute path. If you were to specify `~/ds/config/password`, where ~ refers to the home directory, the server does not expand the ~ value when read from the properties file.

Specifying Default Properties Files

The Directory Server provides a default properties files that apply to all command-line utilities used in client requests. A default properties file, `tools.properties`, is located in the `<server-root>/config` directory.

If you place a custom properties file that has a different filename as `tools.properties` in this default location, you need to specify the path using the --propertiesFilePath option. If you make changes to the `tools.properties` file, you do not need the --propertiesFilePath option. See the examples in the next section.

Evaluation Order Summary

The Directory Server uses the following evaluation ordering to determine options for a given command-line utility:

- All options used with a utility on the command line takes precedence over any options in any properties file.
- If the --propertiesFilePath option is used with no other options, the Directory Server takes its options from the specified properties file.
- If no options are used on the command line including the --propertiesFilePath option (and --noPropertiesFile), the Directory Server searches for the `tools.properties` file at `<server-root>`
- If no default properties file is found and a required option is missing, the tool generates an error.

- Tool-specific properties (for example, `ldapsearch.port=3389`) have precedence over general properties (for example, `port=1389`).

Evaluation Order Example

Given the following properties file that is saved as `<server-root>/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

The Directory Server locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (port and baseDN). The command uses the `bindDN` and `bindPassword` arguments specified in the properties file.

   ```
   $ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
     --propertiesFilePath bin/tools.properties “(objectclass=*)”
   ```
2. Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the Directory Server uses the specified properties file as follows:

   ```
   $ bin/ldapsearch --propertiesFilePath bin/tools.properties \
     “(objectclass=*)”
   ```
3. If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, the Directory Server attempts to locate any default `tools.properties` file in the following location:

   ```
   <server-root>/config/tools.properties
   ```

   Assume that you move your tools.properties file from `<server-root>/bin` to the `<server-root>/config` directory. You can then run your tools as follows:

   ```
   $ bin/ldapsearch "(objectclass=*)"
   ```

   The Directory Server can be configured so that it does not search for a properties file by using the --noPropertiesFile option. This options tells the Directory Server to use only those options specified on the command line. The --propertiesFilePath and --noPropertiesFile options are mutually exclusive and cannot be used together.
4. If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Running Task-based Utilities

The Directory Server has a Tasks subsystem that allows you to schedule basic operations, such as backup, restore, bin/start-server, bin/start-server and others. All task-based utilities require the --task option that explicitly indicates the utility is intended to run as a task rather than in offline mode. The following table shows the arguments that can be used for task-based operations:

Table 71: Task-based Utilities

Option	Description
--task	Indicates that the tool is invoked as a task. The --task argument is required. If a tool is invoked as a task without this --task argument, then a warning message will be displayed stating that it must be used. If the --task argument is provided but the tool was not given the appropriate set of authentication arguments to the

Option	Description
	server, then an error message will be displayed and the tool will exit with an error.
--start	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation starts when scheduled as a server task. A value of '0' causes the task to be scheduled for immediate execution. When this option is used, the operation is scheduled to start at the specified time, after which this utility will exit immediately.
--dependency	Specifies the ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution. This option can be used multiple times in a single command.
--failedDependencyAction	Specifies the action this task will take should one of its dependent tasks fail. The value must be one of the following: `PROCESS`, `CANCEL`, `DISABLE`. If not specified, the default value is `CANCEL`. This option can be used multiple times in a single command.
--completionNotify	Specifies the email address of a recipient to be notified when the task completes. This option can be used multiple times in a single command.
--errorNotify	Specifies the email address of a recipient to be notified if an error occurs when this task executes. This option can be used multiple times in a single command.

PingDirectoryProxy™

Release 7.0

Server Administration Guide

Notice

PingDirectory™ Product Documentation

Trademarks

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Contents

Content-Reference to:

Chapter 1

Introduction

Topics:

PingDirectoryProxy™ Server is a fast and scalable LDAPv3 gateway for the PingDirectory® Server. The Directory Proxy Server architecture can be configured to control how client requests are routed to backend servers.

This chapter provides an overview of the Directory Proxy Server features and components. It contains the following sections:

Overview of the PingDirectoryProxy Server Features

The PingDirectoryProxy Server is a fast, scalable, and easy-to-use LDAP proxy server that provides high availability and additional security for the PingDirectory Server, while remaining largely invisible to client applications. From a client perspective, request processing is the same, whether communicating with the Directory Server directly or going through the Directory Proxy Server.

The PingDirectoryProxy Server provides the following set of features:

- **High availability**. The Directory Proxy Server allows you to transparently fail over between servers if a problem occurs, as well as ensuring that the workload is balanced across the topology. If a client does not support following referrals, the Directory Proxy Server can follow them on the client's behalf.
- **Data mapping and transformation**. The Directory Proxy Server can do DN mapping and attribute mapping to allow clients to interact with the server using older names for directory content. It allows clients to continue working when they would not be able to work directly with the Directory Server, either because of changes that have occurred at the data layer or to inherent design limitations in the clients.
- **Horizontal scalability and performance**. Reads can be horizontally scaled using load balancing. In large data centers, if the data set is too large to be cached or to provide horizontal scalability for writes, the Directory Proxy Server can automatically split the data across multiple systems. This feature allows the Directory Proxy Server to improve scalability and performance of the Directory Server environment.
- **Load balancing and failover**. You can spread the workload across multiple proxies in a large data center using load-balancing algorithms. Load balancing is also useful when a server becomes degraded or non-responsive, because client process requesting is directed to a different server.
- **Security and access control**. The Directory Proxy Server can add additional firewall capabilities, as well as constraints and filtering to help protect the Directory Server from attacks. You can use a Directory Proxy Server in a DMZ as opposed to allowing clients to directly access the Directory Proxy Server in the internal network or providing the data in the DMZ. It can help provide secure access to the data and you can define what actions clients are allowed to do. For example, you can prevent clients from making modifications to data when connected via a VPN no matter what their identity or permissions.
- **Tracking of operations across the environment**. In the past, administrators have commonly complained that when they see a request in the access log, they have no idea where it came from and cannot track it back to a particular client. The Directory Proxy Server contains controls that allow administrators to track requests back to the client that issued them. Whenever the Directory Proxy Server forwards a request to the Directory Server, it includes a control in the request so that the Directory Server's access log has the IP address of the client, address and connection ID of the Directory Server. In the response back to the client, it similarly includes information about the Directory Server that processed the request, such as the connection ID and operation ID. This feature makes it easier for administrators to keep track of what is going on in their environment.
- **Monitoring and management tools**. Because the Directory Proxy Server uses many of the components of the PingDirectory Server, it can leverage them to provide protocol support, logging, management tools for configuration and monitoring, schema, and so on. You can use the Data Metrics Server, the `dsconfig` tool and the Administrative Console to manage the Directory Proxy Server.

Overview of the Directory Proxy Server Components and Terminology

The Directory Proxy Server consists of the following components and functionality that provide the proxy capabilities:

Locations
LDAP External Servers
LDAP Health Checks
Load-Balancing Algorithms
Data Transformations
Request Processors
Server Affinity Providers

Subtree Views
Connection Pools
Client Connection Policies
Entry Balancing

This section describes each component in more detail.

About Locations

Locations define a group of servers with similar response time characteristics. Each location consists of a name and an ordered list of preferred failover locations. The Directory Proxy Server and each of the backend LDAP external servers can be assigned locations. These locations can be taken into account when deciding how to route requests, so that the server prefers to forward requests to Directory Server in the same data center over those in remote locations. As a rule of thumb, if you have multiple data centers then you should have a separate location for each one. In most environments, all Directory Proxy Server instances should have the same configuration except for the attribute that specifies the location of the Directory Proxy Server itself.

For example, a deployment consists of three data centers, one in New York, another in Chicago, and another in Los Angeles. In the New York data center, applications which reside in this data center prefer communicating with directories in this data center. If none of the servers are available, it prefers to failover to the data center in Chicago rather than the data center in Los Angeles. So the New York location contains an ordered list in which the Chicago location is preferred over the Los Angeles data center for failover.

For information about configuring locations, see Configuring Locations.

About LDAP External Servers

You can configure information about the directory server instances accessed by the PingDirectoryProxy Server. This configuration information includes the following:

Server connection information, such as IP address, port, and security layer
Location
Authentication information
Methods for authenticating and authorizing clients
Server-specific health checks
Types of operations allowed. For example, some LDAP external servers may allow only reads and others allow reads and writes, so the Directory Proxy Server can recognize this and accommodate it.

The PingDirectoryProxy Server allows you to configure different types of LDAP external servers. The default configuration for each type is tuned to be the best possible configuration for each.

For information about configuring LDAP external servers, see Configuring LDAP External Servers.

About LDAP Health Checks

The LDAP health check component provides information about the availability of LDAP external servers. The health check result includes a server state, which can be one of the following:

- **Available**. Completely accessible for use.
- **Degraded**. The server may be used if necessary, but has a condition which may make it less desirable than other servers (for example, it is slow to respond or has fallen behind in replication).
- **Unavailable**. Completely unsuitable for use (for example, the server is offline or is missing critical data).

Health check results also include a numeric score, which has a value between 1 and 10, that can help rank servers with the same state. For example, if two servers are available and one has a score of 8 and the other a score of 7, the Directory Proxy Server can be configured to prefer the server with the higher score.

The Directory Proxy Server periodically invokes health checks to monitor each LDAP external server, and may also initiate health checks in response to failed operations. It checks the health of the LDAP external servers at intervals configured in the LDAP server's `health-check-frequency` property. However, the Directory Proxy Server has safeguards in place to ensure that only one health check is in progress at any time against a backend server to avoid affecting its ability to process other requests.

The results of health checks performed by the Directory Proxy Server are made available to the load-balancing algorithms so that they may be taken into account when determining where to send requests. The Directory Proxy Server will attempt to use servers with a state of available before trying servers with a state of degraded. It will never attempt to use servers with a state of unavailable. Some load-balancing algorithms may also take the health check score into account, such as the health-weighted load-balancing algorithm, which prefers servers with higher scores over those with lower scores. You configure the algorithms that work best for you environment.

In some cases, an LDAP health check may define different sets of criteria for promoting and demoting the state of a server. So, a degraded server may need to meet more stringent require- ments to be reclassified as available than it originally took to be considered degraded. For example, if response time is used in the process of determining the health of a server, then the Directory Proxy Server may have a faster response time threshold for transitioning a server from degraded back to available than the threshold used to consider it degraded in the first place. This threshold difference can help avoid cases in which a server repeatedly transitions between the two states because it is operating near the threshold.

For example, the health check used to measure search response time is configured to mark any server to be marked degraded when the search response time is greater than 1 second. You can then configure that the response time must be less than 500 ms before the server is made available again, so that the Directory Proxy Server does not flip back and forth between available and degraded.

PingDirectoryProxy Server provides the following health checks:

- **Measure the response time for searches and examine the entry contents**. For example, the health check might retrieve a monitoring entry from a server and base the health check result on whether the entry was returned, how long it took to be returned, and whether the value of the returned entry matches what was expected.
- **Monitor the replication backlog**. If a server falls too far behind in replication, then the Directory Proxy Server can stop sending requests to it. A server is classified as degraded or unavailable if the threshold is reached for the number of missing changes, the age of the oldest missing change, or both.
- **Consume Directory Server administrative alerts**. If the Directory Server indicates there is a problem, for example an index that must be rebuilt, then it will flag itself as degraded or unavailable. When the Directory Proxy Server detects this, it will stop sending requests to the server. The Directory Proxy Server detects administrative alerts as soon as they are issued by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the Directory Server. When the Directory Proxy Server is notified by the Directory Server of a new alert, it immediately retrieves the base `cn=monitor` entry of the Directory Server. If this entry has a value for the `unavailable-alert-type` attribute, then the Directory Proxy Server will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the Directory Proxy Server will consider it degraded. Clients of the Directory Proxy Server can use a similar mechanism to detect and react when a Directory Proxy Server flags itself as degraded or unavailable.
- **Monitor the busyness of the server**. If a server becomes too busy, then it may be marked degraded or unavailable so that less heavily-loaded servers may be preferred.

For information about configuring health checks, see Configuring Server Health Checks. To associate a health check with an LDAP external server and set the health check frequency, you must configure the `health-check` and `health-check-frequency` properties of the LDAP external server. See “To Configure an External Server Using dsconfig” for information about configuring the properties of the external server.

About Load-Balancing Algorithms

Load-balancing algorithms are used to determine which server in a set of similar servers should be used to process a client request. The algorithm can take the following criteria into account:

- **Consider the location of the server**. Servers in the same location as the Directory Proxy Server can be preferred over those in alternate locations.
- **Consider the health of the server**. Servers that are available are preferred over those that are degraded. In some cases, the health check score may also be used to further differentiate between servers with the same health check state.

- **Route requests consistently**. Requests from a single client may be consistently routed to the same directory server instance to avoid problems such as propagation delay from replication.
- **Retry the operation in an alternate server if the request fails or the operation times out**. You can control if the retry is allowed and, if so, how many times to retry and the time out interval.

The PingDirectoryProxy Server provides the following load-balancing algorithms:

- **Fewest operations**. Requests are forwarded to the backend server with the fewest operations currently in progress.
- **Single server**. Requests are always sent to the same server and will not attempt to fail over to another server if the target server is unavailable.
- **Weighted**. Administrators explicitly assign numeric weights to individual servers or sets of servers to control how likely they are to be selected for processing requests relative to other servers.
- **Health-based weighting**. Uses the health check score to assign weights to each of the servers, so that a server with a higher score gets a higher percentage of the traffic than a server with a lower score. The proportion of traffic received is the difference between their health check scores.
- **Failover**. Requests are always sent to a given server first. If that server fails, then the request is sent to another specified server, and so on through an ordered failover server list.

For information about configuring load balancing, see Configuring Load Balancing.

About Proxy Transformations

Proxy transformations are used to rewrite requests and responses as they pass through the Directory Proxy Server. Proxy data transformations are helpful for clients that use an old schema or that contain a hard-coded schema.

Proxy transformations can provide DN and attribute mapping altering both requests to the server as well as responses from the server. For example, a client sends a request to `o=example.com` even though the directory server handling the request uses `dc=example,dc=com`. The Directory Proxy Server can transparently remap the request so that the server can process it, and map it back to the original DN of the client request when the value is returned. Or if a client tries to use the attribute `userID`, the Directory Proxy Server can map it to `uid` before sending the request on to the backend LDAP server. The Directory Proxy Server then remaps the response to `userID` when the value is returned.

The Directory Proxy Server also includes a proxy transformation that can be used to suppress a specified attribute, so that it will never be returned to clients. It can also cause the server to reject requests which target that particular attribute. Another proxy transformation can be used to prevent entries that match a given search filter from being returned to clients.

For information about configuring proxy transformations, see Configuring Proxy Transformations on page 70.

About Request Processors

A request processor encapsulates the logic for handling an operation, ensuring that a given operation is handled appropriately. The request processor can either process the operation directly, forward the request to another server, or hand off the request to another request processor.

PingDirectoryProxy Server provides the following types of request processor:

- **Proxying request processors**, which forward operations received by the Directory Proxy Server to other LDAP external servers.
- **Entry-balancing request processors**, which split data across multiple servers. They determine which set of servers are used to process a given operation. They then hand off operations to proxying request processors so that requests can be forwarded to one of the servers in the set.
- **Failover request processors**, which perform ordered failover between other types of request processors, sometimes with different behavior for different types of operations.

Directory Proxy Server request processors can be used to forward certain controls, including the batch transaction control and the LDAP join control. The batch transaction control must target a single Berkley DB backend. For more information about the controls, refer to the LDAP SDK for Java documentation.

For information about configuring request processors, see Configuring Request Processors on page 72.

About Server Affinity Providers

The server affinity provider can be used to establish an affinity to a particular backend server for certain operations. You can configure one of three types of provider:

- **Client connection Server Affinity**, so that requests from the same client connection may consistently be routed to the same backend server.
- **Client IP address Server Affinity**, so that all requests coming from the same client system will be consistently routed to the same backend server.
- **Bind DN Server Affinity**, so that all requests from the same user will be consistently routed to the same backend server.

For information about configuring server affinity, see Configuring Server Affinity.

About Subtree Views

A subtree view can be used to make a portion of the DIT available to a client by associating a request processor with a base DN. Subtree views allow you to route operations concerning one set of data to a particular set of data sources, and operations concerning another set of data to another set of data sources. Multiple subtree views may be involved in processing a request, such as for searches that have a scope that is larger than the subtree view.

The subtree view includes a single base DN used to identify the portion of the DIT. They may have hierarchical relationships, for example one subtree view could be configured for `dc=example,dc=com` and another for `ou=People,dc=example,dc=com`.

For information about configuring a subtree view, see Configuring Subtree Views.

About the Connection Pools

Based on the type of backend server that you are using, the PingDirectoryProxy Server maintains either one or two connection pools to the backend server. It maintains either one pool for all types of operations or two separate pools for processing bind and non-bind operations from clients. When the Directory Proxy Server establishes connections, it authenticates them using whatever authentication mechanism is defined in the configuration of the external server. These connections will be re-used for all types of operations to be forwarded to the backend server. The bind DN and password are configured in the Directory Proxy Server.

Whenever a client sends a bind request to the Directory Proxy Server, the server looks at the type of bind request that was sent. If it is a SASL bind request, then the authentication is processed by the Directory Proxy Server itself and it will not be forwarded to the backend server. However, the Directory Proxy Server may use information contained in the backend server as needed. If the bind request is a simple bind request and the bind DN is within the scope of data supplied by the backend server, then the Directory Proxy Server will forward the client request to the backend server so that it will use the credentials provided by the client.

Regardless of the authentication method that the client uses, the Directory Proxy Server will remember the identity of the client after the authentication is complete and for any subsequent requests sent by that client, it will use the configured authorization method to identify the client to the backend server. Even though the operation is forwarded over a connection that is authenticated as a user defined in the Directory Proxy Server configuration, the request is processed by the backend server under the authority of the end client.

About Client Connection Policies

Client connection policies define the general behavior the server exhibits when communicating with a set of clients. Each policy consists of the following:

- A **set of connection criteria** that define which client is associated with the policy based on information the server has about the client, including client address, protocol used, secure communication mechanism, location of the client's entry in the Directory Server and the contents of the client's entry. These criteria are the same as those used for filtered logging. For example, different client connection policies could be established for different classes of users, such as root and non-root users.

- A **set of constraints** on the type of operations a client may request. You can specify whether a particular type of operation is allowed for clients. For some operation types, such as extended operations, you can allow only a particular subset of an operation type, such as a particular extended operation.
- A **set of subtree views** that define information about the parts of the DIT the client may access.

When a client connection is established, only one client connection policy is applied. If the criteria for several policies match the same client connection, the evaluation order index is used as a tiebreaker. If no policy matches, the client connection is terminated. If the client binds, changing its identity, or uses StartTLS to convert from an insecure connection to a secure connection, then the connection may be evaluated again to determine if it matches the same or a different client connection policy. The connection can also be terminated if it no longer matches any policy.

For information about configuring a client connection policy, see Configuring Client Connection Policies on page 77.

About Entry Balancing

Entry balancing allows you to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance. Entry balancing can take advantage of a global index, an in-memory cache used to quickly determine which set or sets of servers should be used to process a request based on the entry DNs and/or the attribute values used in the request.

For information about configuring entry balancing, see *Deploying an Entry-Balancing Proxy Configuration* on page 148.

Server Component Architecture

This section provides an overview of the process flow between the Directory Proxy Server components, for both a simple proxy deployment and an entry-balancing deployment.

Architecture of a Simple Directory Proxy Server Deployment

In a simple Directory Proxy Server deployment, a client request is first processed by a client connection policy as illustrated in Figure 1, "Process Flow for Directory Proxy Server".

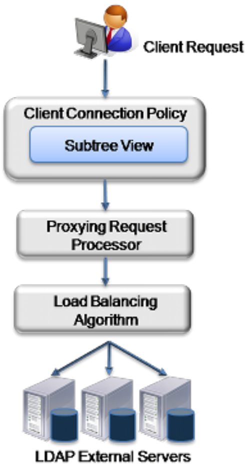


Figure 1: Process Flow for Directory Proxy Server

The client connection policy contains a subtree view, which defines the portion of the DIT available to clients. Once the Directory Proxy Server determines that the DIT is available, it passes the request to the request processor, which defines the logic for processing the request. The request processor then passes the request to a load-balancing algorithm, which determines the server in a set of servers responsible for handling the request. Finally, the request is passed to the LDAP external server. The LDAP external server contains properties that define the server's location in a topology and the health checks used to determine if the server is functioning properly. This information may be used by the load-balancing algorithm in the course of determining how to route requests.

Architecture of an Entry-Balancing Directory Proxy Server Deployment

Figure 2, "Process Flow for Entry-Balancing Directory Proxy Server" describes how a client request is treated in an entry-balancing deployment.

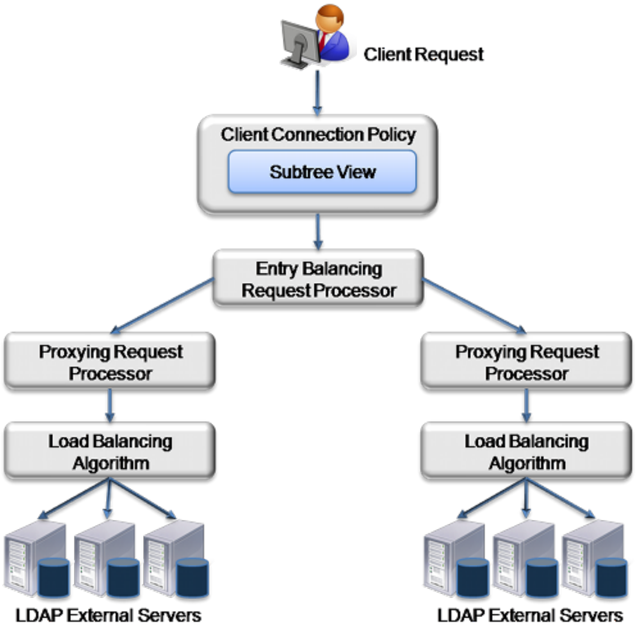


Figure 2: Process Flow for Entry-Balancing Directory Proxy Server

Entry balancing is typically used when the data set it too large to fully cache on a single server or when the write performance requirements of an environment are higher than can be achieved with a single replicated set of servers. In such cases, the data may be split across multiple sets of servers, increasing the memory available for caching and the overall write performance in proportion to the number of server sets.

As with a simple proxy deployment, the client request is first processed by the client connection policy, which determines how the Directory Proxy Server communicates with a set of clients. It contains a subtree view that represents the base DN for the entire deployment. The data set splits beneath this base DN.

The request is then passed to the entry-balancing request processor. The entry-balancing request processor contains a global attribute index property, which helps the request processor determine which server set contains the entry and how to properly route the request. It also contains a placement algorithm, which helps it select the server set in which to place new entries created by add requests.

Beneath the entry-balancing request processor are multiple proxying request processors that handle multiple unique sets of data. These request processors pass the request to a load-balancing algorithm, which determines which LDAP external server should handle the request. As with a simple proxy deployment, this LDAP external server contains properties that define the server's location and the health checks used to determine if the server is functioning properly.

For configuration information, see *Configuring an Entry-Balancing Directory Proxy Server Deployment*. For information about entry-balancing replication, see *Overview of Replication in an Entry-Balancing Environment* on page 170.

Directory Proxy Server Configuration Overview

The configuration of the Directory Proxy Server involves the following steps:

- **Configuring the locations for your deployment**. A location is a collection of servers that share access and latency characteristics. For example, your deployment might include two data centers, one in the east and one in the west. These data centers would be configured as two locations in the Directory Proxy Server. Each location is associated with a name and an ordered list of failover locations, which could be used if none of the servers in the preferred location are available.

- **Configuring the Directory Proxy Server location**. You need to update the configuration to specify the location of the Directory Proxy Server instance.
- **Configuring health checks for the LDAP external servers**. You can configure at what point the Directory Proxy Server considers an LDAP external server to be available, of degraded availability, or unavailable. Each health check can be configured to be used automatically for all LDAP external servers or for a specified set of servers.
- **Configuring the LDAP external servers**. During this step, you define each of the external directory servers, including the server type. You can configure Ping Identity Directory Servers, Sun Java System Directory Servers, or generic LDAP servers. You also assign the server-specific health checks configured in the previous step.
- **Configuring the load-balancing algorithm**. You configure the load-balancing algorithm used by the Directory Proxy Server to determine which server in a set of similar servers should be used to process a client request. The Directory Proxy Server provides default algorithms. It also steps you through the creation of new algorithms by using an existing algorithm as a template or by creating one from scratch.
- **Configuring the proxying request processor**. In this step, you configure proxying request processors that forward operations received by the Directory Proxy Server to other LDAP external servers.
- **Configuring subtree views**. A subtree view defines the portion of the DIT available to a client. Each subtree view can be associated with a load-balancing algorithm to help distribute the work load.
- **Configuring the client connection policy**. You configure policies to classify how different client connections are managed by the Directory Proxy Server. The client connection policy can be used to control the types of operations that a client may perform and the portion of the DIT that the client can access. Restrictions configured in a client connection policy will take precedence over any capabilities granted by access control or privileges.

Chapter

2

Installing the Directory Proxy Server

Topics:

This section describes how to install PingDirectoryProxy Server. It includes pre-installation requirements and considerations.

It includes the following sections:

Before You Begin

The following sections describe requirements and considerations you should make before installing the software and configuring the PingDirectoryProxy Server objects.

Important:

Each Server Deployment Requires an Execution of Setup - Duplicating a Server-root is not Supported. The installation of the server does not write or require any data outside of the server-root directory. After executing `setup`, copying the server-root to another location or system, in order to *duplicate* the installation, is not a supported method of deployment. The server-root can be moved to another host or disk location if a host or file system change is needed.

It is also highly recommended that a Network Time Protocol (NTP) system be in place so that multi-server environments are synchronized and timestamps are accurate.

Defining a Naming Strategy for Server Locations

The various objects defined in the PingDirectoryProxy Server will be specific to a particular location. Location names are used to define a grouping of PingData Server products based on physical proximity. For example, a location is most often associated with a single datacenter location. During the installation, assign a location to each server for optimal inter-server behavior. The location assigned to a server within Global Configuration can be referenced by components within the server as well as processes external to the server to satisfy "local" versus "remote" decisions used in replication, load balancing, and failover.

Software Requirements: Java

For optimized performance, the PingDirectoryProxy Server requires Java for 64-bit architectures. You can view the minimum required Java version on your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

Even if your system already has Java installed, you may want to create a separate Java installation for use by the PingDirectoryProxy Server to ensure that updates to the system-wide Java installation do not inadvertently impact the Directory Proxy Server. This setup requires that the JDK, rather than the JRE, for the 64-bit version, be downloaded.

To Install Java (Oracle/Sun)

1. Open a browser and navigate to the Oracle download site.
2. Download the latest version Java JDK. Click the JDK Download button corresponding to the latest Java update.
3. On the Java JDK page, click the Accept Licence Agreement button, then download the version based on your operating system.

Preparing the Operating System

You should make the following changes to your operating system depending on the production environments on which the PingDirectoryProxy Server will run.

Configuring the File Descriptor Limits

The PingDirectoryProxy Server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. If needed, increase the file descriptor limit on the operating system.

If the operating system relies on `systemd`, refer to the Linux operating system documentation for instructions on setting the file descriptor limit.

To Set the File Descriptor Limit (Linux)

The Directory Proxy Server allows for an unlimited number of connections by default but is restricted by the file descriptor limit on the operating system. Many Linux distributions have a default file descriptor limit of 1024 per process, which may be too low for the server if it needs to handle a large number of concurrent connections.

Once the operating system limit is set, the number of file descriptors that the server will use can be configured by either using a `NUM_FILE_DESCRIPTORS` environment variable, or by creating a `config/num-file-descriptors` file with a single line such as, `NUM_FILE_DESCRIPTORS=12345`. If these are not set, the default of 65535 is used. This is strictly optional if wanting to ensure that the server shuts down safely prior to reaching the file descriptor limit.

1. Display the current hard limit of your system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

   ```
   ulimit -aH
   ```
2. Edit the `/etc/sysctl.conf` file. If there is a line that sets the value of the `fs.file-max` property, make sure its value is set to at least 65535. If there is no line that sets a value for this property, add the following to the end of the file:

   ```
   fs.file-max = 65535
   ```
3. Edit the `/etc/security/limits.conf` file. If the file has lines that sets the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before "#End of file"). Also note that you should insert a tab, rather than spaces, between the columns.

   ```
   * soft nofile 65535
   * hard nofile 65535
   ```
4. Reboot your system, and then use the `ulimit` command to verify that the file descriptor limit is set to 65535.

   ```
   # ulimit -n
   ```

Note: For RedHat 7 or later, modify the `20-nproc.conf` file to set both the open files and max user processes limits:

```
/etc/security/limits.d/20-nproc.conf

 Add or edit the following lines if they do not already exist:

 *          soft    nproc     65536
 *          soft    nofile    65536
 *          hard    nproc     65536
 *          hard    nofile    65536
 root       soft    nproc     unlimited
```

Enabling the Server to Listen on Privileged Ports (Linux)

Linux systems have a mechanism called capabilities that is used to grant specific commands the ability to do things that are normally only allowed for a root account. It may be convenient to enable the server to listen on privileged ports while running as a non-root user.

The `setcap` command is used to assign capabilities to an application. The `cap_net_bind_service` capability enables a service to bind a socket to privileged ports (port numbers less than 1024). If Java is installed in `/ds/java` (and the Java command to run the server is `/ds/java/bin/java`), the Java binary can be granted the `cap_net_bind_service` capability with the following command:

```
$ sudo setcap cap_net_bind_service=+eip /ds/java/bin/java
```

The java binary needs an additional shared library (`libjli.so`) as part of the Java installation. More strict limitations are imposed on where the operating system will look for shared libraries to load for commands that have capabilities assigned. So it is also necessary to tell the operating system where to look for this library. This can be done by creating the file `/etc/ld.so.conf.d/libjli.conf` with the path to the directory that contains the `libjli.so` file. For example, if the Java installation is in `/ds/java`, the contents of that file should be:

```
/ds/java/lib/amd64/jli
```

Run the following command for the change to take effect:

```
$ sudo ldconfig -v
```

To Set the Filesystem Flushes

With the out-of-the-box settings on Linux systems running the ext3 filesystem, the data is only flushed to disk every five seconds. If the Directory Proxy Server is running on a Linux system using the `ext3` filesystem, consider editing the mount options for that filesystem to include the following:

```
commit=1
```

This variable changes the flush frequency from five seconds to one second.

You should also set the flush frequency to the `/etc/fstab` file. Doing the change via the mount command alone will not survive across reboots.

Disable Filesystem Swapping

It is recommended that any performance tuning services like `tuned` be disabled. If performance tuning is required, `vm.swappiness` can be set by cloning the existing performance profile then adding `vm.swappiness = 0` to the new profile's `tuned.conf` file. This file is located at `/usr/lib/tuned/profile-name/tuned.conf`. The updated profile is then selected by running `tuned-adm profile customized_profile`.

About Editing OS-Level Environment Variables

Certain environment variables can impact the Directory Proxy Server in unexpected ways. This is particularly true for environment variables that are used by the underlying operating system to control how it uses non-default libraries.

For this reason, the Directory Proxy Server explicitly overrides the values of key environment variables like *PATH*, *LD_LIBRARY_PATH*, and *LD_PRELOAD* to ensure that something set in the environments that are used to start the server does not inadvertently impact its behavior.

If there is a legitimate need to edit any of these environment variables, the values of those variables should be set by manually editing the `set_environment_vars` function of the `lib/_script-util.sh` script. You will need to stop (bin/stop-server) and re-start (bin/start-server) the server for the change to take effect.

Install sysstat and pstack (Red Hat)

For Red Hat® Linux systems, you should install a couple of packages, `sysstat` and `pstack`, that are disabled by default, but are useful for troubleshooting purposes in the event that a problem occurs. The troubleshooting tool `collect-support-data` uses the `iostat`, `mpstat`, and `pstack` utilities to collect monitoring, performance statistics, and stack trace information on the server's processes. For Red Hat systems, make sure that these packages are installed, for example:

```
$ sudo yum install sysstat gdb dstat -y
```

Install dstat (SUSE Linux)

The `dstat` utility is used by the `collect-support-data` tool and can be obtained from the OpenSuSE project website. The following example shows how to install the `dstat` utility on SuSE Enterprise Linux 11 SP2:

1. Login as Root.
2. Add the appropriate repository using the `zypper` tool.

3. Install the `dstat` utility.

```
$ zypper install dstat
```

Omit vm.overcommit_memory

Administrators should be aware that an improperly configured value for the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file can cause the `setup` or `start-server` tool to fail.

For Linux systems, the `vm.overcommit_memory` property sets the kernel policy for memory allocations. The default value of 0 indicates that the kernel determines the amount of free memory to grant a `malloc` call from an application. If the property is set to a value other than zero, it could lead the operating system to grab too much memory, depriving memory for the `setup` or `start-server` tool.

We recommend omitting the property in the `/etc/sysctl.conf` file to ensure that enough memory is available for these tools.

Managing System Entropy

Entropy is used to calculate random data that is used by the system in cryptographic operations. Some environments with low entropy may have intermittent performance issues with SSL-based communication. This is more typical on virtual machines, but can occur in physical instances as well. Monitor the `kernel.random.entropy_avail` in `sysctl` value for best results.

If necessary, update `$JAVA_HOME/jre/lib/security/java.security` to use `file:/dev/./urandom` for the `securerandom.source` property.

Getting the Installation Packages

To begin the installation process, obtain the latest ZIP release bundle from Ping Identity and unpack it in a folder of your choice. The release bundle contains the Directory Proxy Server code, tools, and package documentation.

To Unpack the Build Distribution

1. Download the latest zip distribution of the Directory Proxy Server software.
2. Unzip the compressed zip archive file in a directory of your choice.

```
$ unzip PingDirectoryProxy-<version>.zip
```

You can now set up the Directory Proxy Server.

Ping License Keys

License keys are required to install all Ping products. Obtain licenses through Salesforce or from https://www.pingidentity.com/en/account/request-license-key.html.

- A license is always required for setting up a new single server instance and can be used site-wide for all servers in an environment. When cloning a server instance with a valid license, no new license is needed.
- A new license must be obtained when updating a server to a new major version, for example from 6.2 to 7.0. Licenses with no expiration date are valid until the server is upgraded to the next major version. A prompt for a new license is displayed during the update process.
- A license may expire on particular date. If a license does expire, obtain a new license and install it using `dsconfig` or the Administrative Console. The server will provide a notification as the expiration date approaches. License details are available using the server's `status` tool.

When installing the server, specify the license key file in one of the following ways:

- Copy the license key file to the server root directory before running setup. The interactive `setup` tool will discover the file and not require input. If the file is not in the server root, the `setup` tool will prompt for its location.
- If the license key is not in the server root directory, specify the `--licenseKeyFile` option for non-interactive setup, and the path to the file.

Installing the Directory Proxy Server

When you deploy PingDirectoryProxy Server in a topology, you generally deploy them in pairs. These pairs are configured identically except for their host name, port name, and possibly their location.

To help administrators easily install identical proxies, the Directory Proxy Server allows you to clone a proxy configuration. First, you install a Directory Proxy Server using the `setup` tool. Then, you configure it using the `create-initial-proxy-config` tool described in Using the create-initial-proxy-config Tool. Finally, you run the `setup` tool on subsequent servers, indicating that you want to clone the configuration on a peer server.

The following sections describe the setup tool in more detail, and tell you how to install first and subsequent proxies in your topology.

About the setup Tool

One of the strengths of the PingDirectoryProxy Server is the ease with which you can install a server instance using the `setup` tool. The `setup` tool allows you to quickly install and configure a stand-alone Directory Proxy Server instance.

To install a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode prompts for information during the installation process. To run the installation in this mode, use the `setup --cli` command.
- **Non-Interactive Command-Line Mode**. Non-interactive command-line mode is designed for setup scripts to automate installations or for command-line usage. To run the installation in this mode, `setup` must be run with the --no-prompt option as well as the other arguments required to define the appropriate initial configuration.

All installation and configuration steps should be performed while logged on to the system as the user or role under which the Directory Proxy Server will run.

Installing the First Directory Proxy Server in Interactive Mode

The `setup` tool provides an interactive text-based interface to install a Directory Proxy Server instance.

To Install the First Directory Proxy Server in Interactive Mode

1. Change to the server root directory.

```
cd Directory Proxy Server
```

2. Use the `setup` command.

```
$ ./setup
```

3. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
4. Press **Enter** to accept the default of `no` in response to adding this new server to an existing topology.

```
Would you like to add this server to an existing Directory Proxy Server
 topology? (yes / no) [no]:
```

5. Enter the fully qualified hostname for this server, or press **Enter** to accept the default.
6. Create the initial root user DN for this server, or press **Enter** to accept the default.
7. Enter and confirm a password for this account.

8. To enable the Directory Proxy Server services (Configuration, Documentation, and SCIM) and Administrative Console over HTTPS, press **Enter** to accept the default. After setup, individual services can be enabled or disabled by configuring the HTTPS Connection Handler.
9. Enter the port on which the Directory Proxy Server should accept connections from HTTPS clients, press **Enter** to accept the default.
10. Enter the port on which the Directory Proxy Server should accept connections from LDAP clients, press **Enter** to accept the default.
11. The next two options enable LDAPS and StartTLS. Press **Enter** to accept the default (yes), or type **no**. If either are enabled, certificate options are required. To use the Java Keystore or the PKCS#12 keystore, the keystore path and the key PIN are required. To use the PKCS#11 token, only the key PIN is required.
12. Choose a certificate server option:

```
Certificate server options:
  1) Generate self-signed certificate (recommended for testing purposes
 only)
  2) Use an existing certificate located on a Java Keystore (JKS)
  3) Use an existing certificate located on a PKCS#12 keystore
  4) Use an existing certificate on a PKCS#11 token
```

13. Choose the desired encryption for backups and log files from the choices provided:

    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.
    - Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
    - Do not encrypt server data.
14. To configure your Directory Proxy Server to use entry balancing, type `yes`, or accept the default `no`. In an entry balancing environment, entries immediately beneath the balancing base DN are divided into disjoint subsets. Each subset of data is handled by a separate set of one or more directory server instances, which replicate this subset of data between themselves. Choosing `yes` will enable more memory be allocated to the server and tools.
15. Choose the option for the amount of memory to assign to this server.
16. Enter an option to setup the server with the current configurtion, provide new parameters, or cancel.
17. Once setup is complete, choose the next configuration option.

```
This server is now ready for configuration What would you like to do?

   1) Start 'create-initial-proxy-config' to create a basic
      initial configuration (recommended for new users)
   2) Start 'dsconfig' to create a configuration from scratch
   3) Quit

Enter choice [1]:
```

To Install Additional Directory Proxy Server Instances in Interactive Mode

The `setup` tool provides an interactive text-based interface to install a Directory Proxy Server instance that clones a previously installed Directory Proxy Server instance.

1. Change to the server root directory.

```
cd Directory Proxy Server
```

2. Use the `setup` command.

```
$ ./setup
```

3. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
4. Enter `yes` in response to add this new server to an existing topology.

```
Would you like to add this server to an existing Directory Proxy Server
 topology? (yes / no) [no]: yes
```

5. Enter the host name of the Directory Proxy Server from which configuration settings are copied during setup.

```
Enter the hostname of the peer Directory Proxy Server from which you would
 like
to copy configuration settings. [proxy.example.com]:
```

6. Type the port number of the peer Directory Proxy Server from which configuration settings are copied during setup. You can press **Enter** to accept the default port, which is 389.

```
Enter the port of the peer Directory Proxy Server [389]:
```

7. Enter the option corresponding to the type of connection you want to use to connect to the peer Directory Proxy Server.

```
How would you like to connect to the peer Directory Proxy Server?
  1) None
  2) SSL
  3) StartTLS

Enter choice [1]:
```

8. Type the root user DN of the peer Directory Proxy Server, or press **Enter** to accept the default (cn=Directory Manager), and then type and confirm the root user password.

```
Enter the manager account DN for the peer Directory Proxy Server
 [cn=Directory Manager]:
Enter the password for cn=Directory Manager:
```

9. Enter the host name of the new local Directory Proxy Server.

```
Enter the fully qualified host name or IP address of the local host
 [proxy.example.com]:
```

10. Choose the location of your new Directory Proxy Server instance or enter a new one.
11. Enter an option to setup the server with the current configurtion, provide new parameters, or cancel.
12. Once setup is complete, choose the next configuration option.

Installing the First Directory Proxy Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. If there is a missing or incorrect argument, the `setup` tool fails and aborts the process.

The `setup` tool automatically chooses the maximum heap size. You can manually tune the maximum amount of memory devoted to the server's process heap using the --maxHeapSize option. The --maxHeapSize argument is only valid if the --entryBalancing or --aggressiveJVMTuning options are also present.

For example, use the --aggressiveJVMTuning option to set the maximum amount of memory used by the Directory Proxy Server and tools as follows:

```
--aggressiveJVMTuning --maxHeapSize 256m
```

If you are using entry balancing, tune the amount of memory devoted to the Directory Proxy Server using the --entryBalancing option as follows:

```
--entryBalancing --maxHeapSize 1g
```

The amount of memory allowed when using the --entryBalancing option is calculated and depends on the amount of system memory available. If you are using entry balancing and also want the tools to get more memory, include both the --entryBalancing and the --aggressiveJVMTuning options.

```
--entryBalancing --aggressiveJVMTuning --maxHeapSize 1g
```

If you have already configured a truststore, you can also use the setup tool to enable security. The following example enables security, both SSL and StartTLS. It also specifies a JKS Keystore and Truststore that define the server certificate and trusted CA. The passwords for the keystore files are defined in the corresponding .pin files, where the password is written on the first line of the file. The values in the .pin files will be copied to the `server-root/config` directory in the `keystore.pin` file.

Note that the password to the private key within the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined within the Administrative Console or `dsconfig` by editing the Trust Manager Provider standard configuration object.

```
$ env JAVA_HOME=/ds/java ./setup --cli \
   --no-prompt --rootUserDN "cn=Directory Manager" \
   --rootUserPassword "password" --ldapPort 389 \
   --enableStartTLS --ldapsPort 636 \
   --useJavaKeystore /path/to/devkeystore.jks \
   --keyStorePasswordFile /path/to/devkeystore.pin \
   --certNickName server-cert \
   --useJavaTrustStore /path/to/devtruststore.jks \
   --trustStorePasswordFile /path/to/devtruststore.pin \
   --acceptLicense
```

To Install the First Directory Proxy Server in Non-Interactive Mode

- Use `setup` with the --no-prompt option. The command uses the default root user DN (`cn=Director Manager`) with the specified --rootUserPassword option. You must include the --acceptLicense option or the `setup` tool will generate an error message.

```
$ env JAVA_HOME=/ds/java ./setup --no-prompt \
--rootUserDN "cn=Directory Manager" \
--rootUserPassword "password" --ldapPort 389 \
--acceptLicense
```

To Install Additional Directory Proxy Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. If there is a missing or incorrect argument, the `setup` tool fails and aborts the process.

To Install Additional Directory Proxy Server in Non-Interactive Mode

- Use `setup` with the --no-prompt option.

```
$ env JAVA_HOME=/ds/java ./setup --cli --no-prompt \
--rootUserDN "cn=Directory Manager" \
--rootUserPassword "password" --ldapPort 1389 \
--localHostName proxy2.example.com \
--peerHostName proxy1.example.com --peerPort 389 \
--peerUseNoSecurity --acceptLicense --location austin1
```

Installing the Directory Proxy Server with a Truststore in Non-Interactive Mode

If you have already configured a trust store, you can also use the `setup` tool to enable security. The following example enables SSL security. It also specifies a JKS Keystore and truststore that define the server certificate and trusted CA. The passwords for the keystore files are defined in the corresponding .pin files, where the password is written on the first line of the file. The values in the .pin files will be copied to the `server-root/config` directory in the `keystore.pin` and `truststore.pin` files.

Note: The password to the private key within the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined within the Administrative Console or `dsconfig` by editing the Key Manager Provider standard configuration object.

To Install the Directory Proxy Server with a Truststore in Non-Interactive Mode

- Run the `setup` tool to install a Directory Proxy Server with a truststore.

```
$ env JAVA_HOME=/ds/java ./setup --cli \
  --no-prompt --rootUserDN "cn=Directory Manager" \
  --rootUserPassword "password" \
  --ldapPort 389 --ldapsPort 636 \
  --useJavaKeystore /path/to/devkeystore.jks \
  --keyStorePasswordFile /path/to/devkeystore.pin \
  --certNickName server-cert \
  --useJavaTrustStore /path/to/devtruststore.jks \
  --acceptLicense

In order to update the trust store, the password must be provided

See 'prepare-external-server --help' for general overview

Testing connection to ds-east-01.example.com:1636 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access .....
Created 'cn=Proxy User,cn=Root DNs,cn=config'

Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

About the Layout of the Directory Proxy Server Folders

Once you have unzipped the Directory Proxy Server distribution file, the following folders and command-line utilities are available.

Table 1: Layout of the Directory Proxy Server Folders

Directories/Files/Tools	Description
License.txt	Licensing agreement for the Directory Proxy Server.
README	README file that describes the steps to set up and start the Directory Proxy Server.
bak	Stores the physical backup files used with the `backup` command-line tool.
bat	Stores Windows-based command-line tools for the Directory Proxy Server.
bin	Stores UNIX/Linux-based command-line tools for the Directory Proxy Server.
classes	Stores any external classes for server extensions.
collector	Used by the server to make monitored statistics available to the Data Metrics Server.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
docs	Provides the product documentation.
import-tmp	Stores temporary imported items.
ldif	Stores any LDIF files that you may have created or imported.

Directories/Files/Tools	Description
legal-notices	Stores any legal notices for dependent software used with the Directory Proxy Server.
lib	Stores any scripts, jar, and library files needed for the server and its extensions.
locks	Stores any lock files in the backends.
logs	Stores log files for the Directory Proxy Server.
metrics	Stores the metrics that can be gathered for this server and surfaced in the Data Metrics Server.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
revert-update	The `revert-update` tool for UNIX/Linux systems.
revert-update.bat	The `revert-update` tool for Windows systems.
setup	The `setup` tool for UNIX/Linux systems.
setup.bat	The `setup` tool for Windows systems.
scim-data-tmp	Used to create temporary files containing SCIM request data.
uninstall	The `uninstall` tool for UNIX/Linux systems.
uninstall.bat	The `uninstall` tool for Windows systems.
update	The `update` tool for UNIX/Linux systems.
update.bat	The `update` tool for Windows systems.
Velocity	Stores any customized Velocity templates and other artifacts (CSS, Javascript, images), or Velocity applications hosted by the server.

Running the Server

To start the Directory Proxy Server, run the bin/start-server command on UNIX or Linux systems (an analogous command is in the `bat` folder on Microsoft Windows systems). The bin/start-server command starts the Directory Proxy Server as a background process when no options are specified. To run the Directory Proxy Server as a foreground process, use the bin/start-server command with the --nodetach option.

To Start the Directory Proxy Server

Use bin/start-server to start the server.

```
$ bin/start-server
```

To Run the Server as a Foreground Process

1. Enter bin/start-server with the --nodetach option to launch the Directory Proxy Server as a foreground process.

```
$ bin/start-server --nodetach
```

2. You can stop the Directory Proxy Server by pressing `CNTRL+C` in the terminal window where the server is running or by running the bin/stop-server command from another window.

To Start the Server at Boot Time

By default, the PingDirectoryProxy Server does not start automatically when the system is booted. Instead, you must manually start it with the bin/start-server command. To configure the Directory Proxy Server to start automatically when the system boots, use the `create-systemd-script` utility to create a script, or create the script manually.

1. Create the service unit configuration file in a temporary location where "ds" is the user the PingDirectoryProxy will run as.

   ```
   $ bin/create-systemd-script \
        --outputFile /tmp/ping-directory.service \
        --userName ds
   ```
2. As a root user, copy the `ping-directory.service` configuration file into the `/etc/systemd/system` directory.
3. Reload `systemd` to read the new configuration file.

   ```
   $ systemctl daemon-reload
   ```
4. To start the PingDirectoryProxy, use the `start` command.

   ```
   $ systemctl start ping-directory.service
   ```
5. To configure the PingDirectoryProxy to start automatically when the system boots, use the `enable` command.

   ```
   $ systemctl enable ping-directory.service
   ```
6. Log out as root.

   If on an RC system, this task is done by creating the startup script with `bin/create-rc-script` and moving it to the `/etc/init.d` directory. Create symlinks to it from the `/etc/rc3.d` directory (staring with an “S” to ensure that the server is started) and `/etc/rc0.d` directory (starting with a “K” to ensure that the server is stopped).

Logging into the Administrative Console

After the server is installed, access the Administrative Console, `https://hostname:HTTPport/console/login`, to verify the configuration and manage the server. To log into the Administrative Console, use the initial root user DN specified during setup (by default `cn=Directory Manager`).

The `dsconfig` command or the Administrative Console can be used to create additional root DN users in `cn=Root DNs,cn=config`. These new users require the fully qualified DN as the login name, such as `cn=new-admin,cn=Root DNs,cn=config`. To use a simple user name (with out the `cn=` prefix) for logging into the Administrative Console, the root DN user must have the `alternate-bind-dn` attribute configured with an alternate name, such as "admin."

By default the link to the Administrative Console is `https://hostname:HTTPport/console/login`.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package (`/server-root/resource/admin-console.zip`) can be installed according to that container's documentation.

Stopping the Directory Proxy Server

The Directory Proxy Server provides a simple shutdown script, bin/stop-server, to stop the server. You can run it manually from the command line or within a script.

If the Directory Proxy Server has been configured to use a large amount of memory, then it can take several seconds for the operating system to fully release the memory and make it available again. If you try to start the server too quickly after shutting it down, then the server can fail because the system does not yet have enough free memory. On UNIX systems, run the `vmstat` command and watch the values in the "free" column increase until all memory held by the Directory Proxy Server is released back to the system.

You can also set a configuration option that specifies the maximum shutdown time a process may take.

To Stop the Server

- Use the bin/stop-server tool to shut down the server.

```
$ bin/stop-server
```

To Schedule a Server Shutdown

- Use the `bin/stop-server` tool with the --stopTime YYYYMMDDhhmmss option to schedule a server shutdown.

  The Directory Proxy Server schedules the shutdown and sends a notification to the `server.out` log file. The following example sets up a shutdown task that is scheduled to be processed on June 6, 2012 at 8:45 A.M. CDT. The server uses the UTC time format if the provided timestamp includes a trailing “Z”, for example, 20120606134500Z. The command also uses the --stopReason option that writes the reason for the shut down to the logs.

```
$ bin/stop-server --stopTime 20120606134500Z --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --stopReason "Scheduled offline maintenance"
```

To Restart the Server

Re-start the Directory Proxy Server using the bin/stop-server command with the `--restart` or `-R` option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again.

- Go to the server root directory, and run the bin/stop-server command with the -R or --restart options.

```
$ bin/stop-server --restart
```

Run the Server as a Microsoft Windows Service

The server can run as a Windows service on Windows Server 2012 R2 and Windows Server 2016. This enables log out of a machine without the server being stopped.

To Register the Server as a Windows Service

Perform the following steps to register the server as a service:

1. Stop the server with `bin/stop-server`. A server cannot be registered while it is running.
2. Register the server as a service. From a Windows command prompt, run `bat/register-windows-service.bat`.
3. After a server is registered, start the server from the Windows Services Control Panel or with the `bat/start-server.bat` command.

   Command-line arguments for the `start-server.bat` and `stop-server.bat` scripts are not supported while the server is registered to run as a Windows service. Using a task to stop the server is also not supported.

To Run Multiple Service Instances

Only one instance of a particular service can run at one time. Services are distinguished by the `wrapper.name` property in the `<server-root>/config/wrapper-product.conf` file. To run additional service instances, change the `wrapper.name` property on each additional instance. Descriptions of the services can also be added or changed in the `wrapper-product.conf` file.

To Deregister and Uninstall Services

While a server is registered as a service, it cannot run as a non-service process or be uninstalled. Use the `bat/deregister-windows-service.bat` file to remove the service from the Windows registry. The server can then be uninstalled with the `uninstall.bat` script.

Log Files for Services

The log files are stored in `<server-root>/logs`, and filenames start with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or due to file size. Only the last three log files are retained. These configurations can be changed in the `<server-root>/config/wrapper.conf` file.

Uninstalling the Server

The Directory Proxy Server provides an `uninstall` command-line utility for quick and easy removal of the code base.

To uninstall a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode is a text-based interface that prompts the user for input. You can start the command using the `bin/uninstall` command with the --cli option. The utility prompts you for input if more data is required.
- **Non-Interactive Command-Line Mode**. Non-interactive mode suppresses progress information from being written to standard output during processing, except for fatal errors. This mode is convenient for scripting and is invoked using the `bin/uninstall` command with the --no-prompt option.

Note: For stand-alone installations with a single Directory Proxy Server instance, you can also manually remove the Directory Proxy Server by stopping the server and recursively deleting the directory and subdirectories. For example:

```
$ rm -rf /ds/PingDirectoryProxy
```

To Uninstall the Server in Interactive Mode

Interactive mode uses a text-based, command-line interface to help you remove your instance. If `uninstall` cannot remove all of the Directory Proxy Server files, the `uninstall` tool generates a message with a list of the files and directories that must be manually deleted. The `uninstall` command must be run as either the root user or the same user (or role) that installed the Directory Proxy Server.

1. From the server root directory, run the `uninstall` command.

```
$ ./uninstall --cli
```

2. Select the components to be removed. If you want to remove all components, press **Enter** to accept the default (remove all). Enter the option to specify the specific components that you want to remove.

```
Do you want to remove all components or select the components to remove?

1) Remove all components
2) Select the components to be removed

q) quit
Enter choice [1]:
```

3. For each type of server component, press **Enter** to remove them or type `no` to keep it.

```
Remove Server Libraries and Administrative Tools? (yes / no) [yes]:
Remove Database Contents? (yes / no) [yes]:
Remove Log Files? (yes / no) [yes]:
Remove Configuration and Schema Files? (yes / no) [yes]:
Remove Backup Files Contained in bak Directory? (yes / no) [yes]:
Remove LDIF Export Files Contained in ldif Directory? (yes / no) [yes]:
```

4. If the Directory Proxy Server is part of a replication topology, type `yes` to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a stand-alone server, continue to step 7.

5. Type the Global Administrator ID and password to remove the references to this server in other replicated servers. Then, type or verify the host name or IP address for the server that you are uninstalling.
6. Next, select how you want to trust the server certificate if you have set up SSL or StartTLS. For this example, press **Enter** to accept the default.

```
How do you want to trust the server certificate for the Directory Proxy
 Server
on server.example.com:389?

1) Automatically trust
2) Use a trust store
3) Manually validate

Enter choice [3]:
```

7. If your Directory Proxy Server is running, the server is shutdown before continuing the uninstall process. The uninstall processes the removal requests and completes. View the logs for any remaining files. Manually remove any remaining files or directories, if listed.

To Uninstall the Server in Non-Interactive Mode

The `uninstall` utility provides a non-interactive method to enter the command with the --no-prompt option. Another useful argument is the --forceOnError option that continues the uninstall process when an error is encountered. If an option is incorrectly entered or if a required option is omitted and the --forceOnError option is not used, the command will fail and abort.

1. From the server root directory, run `uninstall` tool with the --remove-all option to remove all of the Directory Proxy Server's libraries. The --quiet option suppresses output information and is optional. The following command assumes that the Directory Proxy Server is stand-alone and not part of a replication topology.

```
$ ./uninstall --cli --remove-all --no-prompt --quiet --forceOnError
```

2. If any files or directories remain, manually remove them.

To Uninstall Selected Components in Non-Interactive Mode

From the server root directory, run `uninstall` with the --backup-files option to remove the Directory Proxy Server's backup files. Use the --help or -H option to view the other options available to remove specific components.

```
$ ./uninstall --cli --backup-files --no-prompt --quiet --forceOnError
```

Updating the Directory Proxy Server

Ping Identity issues new software builds periodically and distributes the software package in zip format. Administrators can use the Directory Proxy Server's `update` utility to update the current server code with the latest features and bug fixes. To update the Directory Proxy Server to a newer version, download the build package, and then unzip the new server package on the same host as the server that you wish to update. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors.

During an update process, the updater checks a manifest file that contains a MD5 checksum of each file in its original state when installed from zip. Next, it compares the checksum of the new server files to that of the old server. Any files that have different checksums will be updated. For files that predates the manifest file generation, the file is backed up and replaced. The updater also logs all file changes in the history directory to tell what files have been changed.

For schema updates, the `update` tool preserves any custom schema definitions (`99-user.ldif`). For any default schema element changes, if any, the updater will warn the user about this condition and then create a patch schema file and copy it into the server's schema directory. For configuration files, the update tool preserves the configuration file, `config.ldif`, unless new configuration options must be added to the Directory Proxy Server.

Once the updater finishes its processing, it checks if the newly updated server starts without any fatal errors. If an error occurs during the update process, the `update` tool reverts the server root instance to the server state prior to the update.

About Updating Servers in a Topology

An update to the current release includes significant changes, and the introduction of a topology registry, which will store information previously stored in the admin backend (server instances, instance and secret keys, server groups, and administrator user accounts). For the admin backend to be migrated, the `update` tool must be provided LDAP authentication options to the peer servers of the server being updated.

The LDAP connection security option requested (either plain, TLS, StartTLS, or SASL) must be configured on every server in the topology. The LDAP credentials must be present on every server in the topology, and must have permissions to read from the admin backend and the config backend of every server in the topology. For example, a root DN user with the `inherit-default-privileges` set to true (such as the `cn=Directory Manager` user) that exists on every server can be used.

After enabling or fixing the configuration of the LDAP connection handler(s) to support the desired connection security mechanism on each server, run the following `dsframework` command on the server being updated so that its admin backend has the most up-to-date information:

```
$ bin/dsframework set-server-properties \
  --serverID serverID \
  --set ldapport:port \
  --set ldapsport:port \
  --set startTLSEnabled:true
```

The `update` tool will verify that the following conditions are satisfied on every server in the topology before allowing the update:

- When the first server is being updated, all other servers in the topology must be online. When updating additional servers, all topology information will be obtained from one of the servers that has already been updated. The `update` tool will connect to the peer servers of the server being updated to obtain the necessary information to populate the topology registry. The provided LDAP credentials must have read permissions to the config and admin backends of the peer servers.
- The instance name is set on every server, and is unique across all servers in the topology. The instance name is a server's identifier in the topology. After all servers in the topology have been updated, each server will be uniquely identified by its instance name. Once set, the name cannot be changed. If needed, the following command can be used to set the instance name of a server prior to the update:

  ```
  $ bin/dsconfig set-global-configuration-prop \
    --set instance-name:uniqueName
  ```
- The cluster-wide configuration is synchronized on all servers in the topology. Older versions have some topology configuration under `cn=cluster,cn=config` (JSON attribute and field constraints). These items did not support mirrored cluster-wide configuration data. An update should avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. To synchronize the cluster-wide configuration data across all servers in the topology, run the `config-diff` tool on each pair of servers to determine the differences, and use `dsconfig` to update each instance using the `config-diff` output. For example:

  ```
  $ bin/config-diff --sourceHost hostName \
    --sourcePort port \
    --sourceBindDN bindDN \
    --sourceBindPassword password \
    --targetHost hostName \
    --targetPort port \
    --targetBindDN bindDN \
    --targetBindPassword password
  ```

If any of these conditions are not satisfied, the `update` tool will list all of the errors encountered for each server, and provide instructions on how to fix them.

To Update the Directory Proxy Server

Assume that an existing version of the Directory Proxy Server is stored at PingDirectoryProxy-old, which you want to update.

1. Make sure you have complete, readable backup of the existing system before upgrading the Directory Proxy Server build. Also, make sure you have a clear backout plan and schedule.
2. Download the latest version of the PingDirectoryProxy Server software and unzip the file. For this example, let's assume the new server is located in the PingDirectoryProxy-new directory.
3. Check the version number of the newly downloaded Directory Proxy Server instance using the --version option on any command-line utility. For example, you should see the latest revision number.

```
$ PingDirectoryProxy-new/setup --version PingDirectoryProxy Server 7.0.0.0
Build 20110432006092 Revision 9235
```

4. Use the `update` tool of the newly unzipped build to update the Directory Proxy Server code. Make sure to specify the Directory Proxy Server instance that you are upgrading with the --serverRoot option. The Directory Proxy Server must be stopped for this update to be applied.

```
$ PingDirectoryProxy-new/update --serverRoot PingDirectoryProxy-old
```

Note: The PingDirectoryProxy Server provides a web console called the Administrative Console, to configure and monitor the server. If you update the Directory Proxy Server version, you should also update the Administrative Console.

5. View the log file to see which files were changed. The log file is located in the `<server-root>/history` directory. For example, the file will be labelled with the Directory Proxy Server version number and revision.

```
$ view <server-root>/history/1272307020420-7.0.0.0.9235/update.log
```

Reverting an Update

Once the Directory Proxy Server has been updated, you can revert to the most recent version (one level back) using the `revert-update` tool. The `revert-update` tool accesses a log of file actions taken by the updater to put the filesystem back to its prior state. If you have run multiple updates, you can run the `revert-update` tool multiple times to revert to each prior update sequentially. You can only revert back one level. For example, if you have run the update twice since first installing the Directory Proxy Server, you can run the `revert-update` command to revert to its previous state, then run the `revert-update` command again to return to its original state.

Note: Reverting an update is not supported for upgrades to version 7.0, due to the topology backend changes.

To Revert to the Most Recent Server Version

Use `revert-update` in the server root directory revert back to the most recent version of the server.

```
$ PingDirectoryProxy-old/revert-update
```

Configure SCIM After Upgrade

Modifications in SCIM PATCH are mapped directly to LDAP modifications to use the matching rules configured in the Directory Proxy Server, when matching deleted values. Since the SCIM PATCH is now applied by the Directory Server, the Permissive Modify Request Control (1.2.840.113556.1.4.1413) is now required by the SCIM component. This ensures that adding an existing value or deleting a non-existent value in the PATCH request will not generate an error. This affects upgrades from server versions prior to 5.0.0.

To continue using the SCIM component after an upgrade, access controls and configuration must be updated to allow access to the Permissive Modify Request Control. Run the `dsconfig` commands to update these components:

```
$ dsconfig set-access-control-handler-prop \
  --remove 'global-aci:(targetcontrol="1.3.6.1.1.13.2 ||
 1.2.840.113556.1.4.473 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9
 || 1.3.6.1.1.12")(version 3.0;acl "Authenticated access to controls used by
 the SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473
 || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9 || 1.3.6.1.1.12 ||
 1.2.840.113556.1.4.1413")(version 3.0;acl "Authenticated access to controls
 used by the SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

```
dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add supported-control-oid:1.2.840.113556.1.4.1413
```

In the last command, `dc_example_dc_com-req-processor` is the default processor name. Replace it with the correct name for your system.

Chapter

3

Configuring the Directory Proxy Server

Topics:

Once you have initially configured the PingDirectoryProxy Server, you can manage your deployment using the configuration framework and management tools. This chapter briefly describes these tools and provides procedures to help you maintain and update your deployment.

It includes the following sections:

- *Configuring the Global Referential Integrity Plug-in*
- *Configuring an Active Directory Server Back-end*

About the Configuration Tools

The PingDirectoryProxy Server configuration can be accessed and modified in the following ways:

- **Using the Administrative Console**. The PingDirectoryProxy Server provides an Administrative Console for graphical server management and monitoring. The console provides equivalent functionality as the `dsconfig` command for viewing or editing configurations. All configuration changes using this tool are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands should you need to back out of a configuration.
- **Using the dsconfig Command-Line Tool**. The `dsconfig` tool is a text-based menu-driven interface to the underlying configuration. The tool runs the configuration using three operational modes: interactive command-line mode, non-interactive command-line mode, and batch mode. All configuration changes made using this tool are recorded in `logs/config-audit.log`.

Using the create-initial-proxy-config Tool

The `create-initial-proxy-config` tool can be used to initially configure the Directory Proxy Server. We strongly recommend that you use the `create-initial-proxy-config` tool for your initial Directory Proxy Server configuration. This tool prompts you for basic information about your topology, including external servers, their locations, and credentials for communicating with them. Once the configuration is complete, the tool writes the configuration to a `dsconfig` batch file and allows you to apply the configuration to the local Directory Proxy Server. The tool assumes the following about your topology:

- All servers are accessible through a single user account. This user account must be a root user that is not generally accessible to clients to avoid inadvertent changes, deletions, or backend server availability issues due to reimporting data.
- All servers support the same type of communication security.
- All external servers are any combination of Ping Identity Directory Server, Sun Directory Server, or Red Hat (including Fedora and 389) instances.

If your topology does have these characteristics, you can use the tool to define a basic configuration that is saved to a `dsconfig` batch file. You can then run the `dsconfig` tool to fine-tune the configuration. You can also use this tool to configure an entry balancing configuration, which allows you to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance.

The `create-initial-proxy-config` tool produces a log file called `create-initial-proxy-config.log` that is stored in the local Directory Proxy Server's `logs` directory.

You can only run the `create-initial-proxy-config` tool once for the initial configuration of each Directory Proxy Server instance. To tune your configuration, use the `dsconfig` tool. When installing a second Directory Proxy Server, it will not be necessary to run the `create-initial-proxy-config` tool again, as the Directory Proxy Server setup has the ability to clone the settings from an existing Directory Proxy Server.

This section describes how to use this tool to configure a standard Directory Proxy Server deployment as well as an entry balancing configuration.

Configuring a Standard Directory Proxy Server Deployment

This section describes how to install a standard Directory Proxy Server deployment using the `create-initial-proxy-config` tool. Remember that you deploy the Directory Proxy Server in pairs. Each pair should be configured identically except for their host name, port, and possibly their location.

To Configure a Standard Directory Proxy Server Deployment

1. After initial installation, select the number to start the `create-initial-proxy-config` tool automatically. Otherwise, run it manually at the command line from the server root directory, <server-root>/PingDirectoryProxy.

```
$ ./bin/create-initial-proxy-config
```

2. The initial proxy configuration presents the assumptions about the underlying Directory Server backend servers. If the servers do not meet the requirements, then you can enter "no" to quit the process.

```
Some assumptions are made about the topology in order to keep this tool
 simple:

 1) all servers will be accessible via a single user account
 2) all servers support the same communication security type
 3) all servers are PingData, Alcatel-Lucent 8661, Sun
    Java System 5.x, 6.x, or 7.x, or Red Hat (including Fedora and 389)
    directory servers

If your topology does not have these characteristics you can use this tool
 to define a basic configuration and then use the 'dsconfig' tool or the
 Administrative Console to
fine tune the configuration.

Continue? (yes / no) [yes]:
```

3. Enter the DN for the Directory Proxy Server user account, then enter and confirm the password for this account. Note that you should not use `cn=Directory Manager` account for communication between the Directory Proxy Server and the Directory Server. For security reasons, the account used to communicate between the Directory Proxy Server and the Directory Server should not be directly accessible by clients accessing the Directory Proxy Server. For more information about this account, see Configuring LDAP External Servers.

```
Enter the DN of the proxy user account [cn=Proxy User,cn=Root
 DNs,cn=config]:
Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

4. Specify whether you will be using secure communication with the Directory Server instances.

```
>>>> External Server Communication Security

Specify the type of security that the Directory Proxy Server will use when
 communicating with directory server instances:

   1)  None
   2)  SSL
   3)  StartTLS

   b)  back
   q)  quit

Enter choice [1]:
```

5. Specify the base DNs of the Directory Server instances that will be accessed through the Directory Proxy Server. The Directory Proxy Server will create subtree views using each base DN to define portions of the external servers' DIT available for client access. You can specify more than one base DN. Press **Enter** when you have finished specifying the DN(s).

```
Enter a base DN of the directory server instances that will be accessed
 through the Identity Proxy:

    b)  back
    q)  quit

Enter a DN or choose a menu item [dc=example,dc=com]:
```

6. Next, specify if the entries under your defined subtree view will be split across multiple servers in an entry balanced deployment. For this example, press Enter to accept the default ("no").

7. Define a location for your server, such as the name of your data center or the city where the server is located. This example illustrates defining a location named `east`.

```
Enter a location name or choose a menu item: east
```

8. If you defined more than one location, specify the location that contains the Directory Proxy Server itself.

```
Choose the location for this Directory Proxy Server

    1)  east
    2)  west

    b)  back
    q)  quit

Enter choice [1]: 1
```

9. Define the hostname:port used by the LDAP external servers. If you have specified more than one location, you will go through this process for each location.

```
Enter a host:port or choose a menu item [localhost:389]: ldap-
east-01.example.com:389
```

10. After each step, the server will attempt to prepare each external server by testing the communication between the Directory Proxy Server and the Directory Server. Select the option "Yes, and all subsequent servers" to indicate that you want the tool to create a proxy user account on all of your LDAP external servers within that location.

```
Would you like to prepare ldap-east-01.example.com:389 for access by the
 Directory Proxy Server?

    1) Yes
    2) No
    3) Yes, and all subsequent servers
    4) No, and all subsequent servers

Enter choice [1]: 3
```

11. If the proxy user account did not previously exist on your LDAP external server, create the account by connecting as `cn=Directory Manager`.

```
Would you like to create or modify root user 'cn=Proxy User' so that it is
 available for this Directory Proxy Server? (yes / no) [yes]:
Enter the DN of an account on ldap-east-01.example.com:389 with which to
 create or manage the 'cn=Proxy
User' account [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

12. Repeat steps 9-12 for the servers in the other location. Then, press **Enter** to finish configuring the location.

13. Review the configuration summary. Once you have confirmed that the changes are correct, press **Enter** to write the configuration.

```
>>>> Configuration Summary

  External Server Security:  SSL
  Proxy User DN:             cn=Proxy User,cn=Root DNs,cn=config

  Location east
    Failover Order: west
    Servers: localhost:1636
  Location west
    Failover Order: east
    Servers: localhost:2636

  Base DN: dc=example,dc=com
    Servers: localhost:1636, localhost:2636


    b)  back
    q)  quit
    w)  write configuration file

Enter choice [w]:
```

14. Next, apply the configuration changes locally to the Directory Proxy Server. If you have any Server SDK extensions, make sure to run the `manage-extension` tool, then press **Enter** to apply the changes to the Directory Proxy Server. Alternatively, you can quit and instead run the `dsconfig` batch file at a later time. Once the changes have been applied, you cannot use the `create-initial-proxy-config` tool to configure this Directory Proxy Server again. Instead, use the `dsconfig` tool.

```
This tool can apply the configuration changes to the local Identity Proxy.
 This requires any configured Server SDK extensions to be in place. Do you
 want to do
this? (yes / no) [yes]:
```

If you open the generated `proxy-cfg.txt` file or the `logs/config-audit.log` file, you will see that a configuration element hierarchy has been created: locations, health checks, external servers, load-balancing algorithms, request processors, and subtree views.

About dsconfig Configuration Tool

The `dsconfig` tool is the text-based management tool used to configure the underlying Directory Server configuration. The tool has three operational modes: interactive mode, non-interactive mode, and batch mode.

The `dsconfig` tool also offers an offline mode using the --offline option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Using dsconfig in Interactive Command-Line Mode

In interactive mode, the `dsconfig` tool offers a filtering mechanism that only displays the most common configuration elements. The user can specify that more expert level objects and configuration properties be shown using the menu system.

Running `dsconfig` in interactive command-line mode provides a user-friendly, menu-driven interface for accessing and configuring the PingDirectoryProxy Server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` script without any arguments. You will be prompted for connection and authentication information to the Directory Proxy Server, and then a menu will be displayed of the available operation types.

In some cases, a default value will be provided in square brackets. For example, [389] indicates that the default value for that field is port 389. You can press **Enter** to accept the default. To skip the connection and authentication prompts, provide this information using the command-line options of `dsconfig`.

Using dsconfig Interactive Mode: Viewing Object Menus

Because some configuration objects are more likely to be modified than others, the PingDirectoryProxy Server provides four different object menus that hide or expose configuration objects to the user. The purpose of object levels is to simply present only those properties that an administrator will likely use. The Object type is a convenience feature designed to unclutter menu readability.

The following object menus are available:

- **Basic**. Only includes the components that are expected to be configured most frequently.
- **Standard**. Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.
- **Advanced**. Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be potentially harmful if configured incorrectly.
- **Expert**. Includes all components in the Basic, Standard, and Advanced menus plus other components that should almost never require configuration or that could seriously impact the functionality of the server if not properly configured.

To Change the dsconfig Object Menu

1. Repeat steps 1–6 in the section using `dsconfig` in To Install the Directory Proxy Server in Interactive Mode.
2. On the **PingDirectoryProxy Server configuration** main menu, type **o** (letter “o”) to change the object level. By default, Basic objects are displayed.
3. Enter a number corresponding to a object level of your choice: `1` for Basic, `2` for Standard, `3` for Advanced, `4` for Expert.
4. View the menu at the new object level. Additional configuration options for the Directory Proxy Server components are displayed.

Using dsconfig in Non-Interactive Mode

The `dsconfig` non-interactive command-line mode provides a simple way to make arbitrary changes to the Directory Proxy Server by invoking it from the command line. To use administrative scripts to automate configuration changes, run the `dsconfig` command in non-interactive mode, which is convenient scripting applications. Note, however, that if you plan to make changes to multiple configuration objects at the same time, then the batch mode might be more appropriate.

You can use the `dsconfig` tool to update a single configuration object using command-line arguments to provide all of the necessary information. The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The --no-prompt argument indicates that you want to use non-interactive mode. The {sub-command} is used to indicate which general action to perform. The {globalArgs} argument provides a set of arguments that specify how to connect and authenticate to the Directory Proxy Server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on your setup. For example, using standard LDAP connections, you can invoke the `dsconfig` tool as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

If your system uses SASL GSSAPI (Kerberos), you can invoke `dsconfig` as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The {subcommandArgs} argument contains a set of arguments specific to the particular subcommand that you wish to invoke. To always display the advanced properties, use the --advanced command-line option.

Note: Global arguments can appear anywhere on the command line (including before the subcommand, and after or intermingled with subcommand-specific arguments). The subcommand-specific arguments can appear anywhere after the subcommand.

To Get the Equivalent dsconfig Non-Interactive Mode Command

1. Using `dsconfig` in interactive mode, make changes to a configuration but do not apply the changes (that is, do not enter "f").
2. Enter `d` to view the equivalent non-interactive command.
3. View the equivalent command (seen below), and then press **Enter** to continue. For example, based on an example in the previous section, changes made to the `db-cache-percent` returns the following:

   ```
   Command line to apply pending changes to this Local DB Backend:
   dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:40
   ```

   The command does not contain the LDAP connection parameters required for the tool to connect to the host since it is presumed that the command would be used to connect to a different remote host.

Using dsconfig Batch Mode

The PingDirectoryProxy Server provides a `dsconfig` batching mechanism that reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The --no-prompt option is required with `dsconfig` in batch mode.

If a `dsconfig` command has a missing or incorrect argument, the command will fail and abort the batch process without applying any changes to the Directory Proxy Server. The `dsconfig` command supports a `--batch-continue-on-error` option which instructs `dsconfig` to apply all changes and skip any errors.

You can view the `logs/config-audit.log` file to review the configuration changes made to the Directory Proxy Server and use them in the batch file. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The Directory Proxy Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to PingDirectoryProxy Server machines.

To Configure the Directory Proxy Server in dsconfig Batch Mode

1. Create a text file that lists each `dsconfig` command with the complete set of properties that you want to apply to the Directory Proxy Server. The items in this file should be in the same format as those accepted by the `dsconfig` command. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

   ```
   # This dsconfig operation creates the exAccountNumber global attribute
    index.
   dsconfig create-global-attribute-index
   --processor-name ou_people_dc_example_dc_com-eb-req-processor
   --index-name exAccountNumber --set prime-index:true

   # Here we create the entry-count placement algorithm with the
   # default behavior of adding entries to the smallest backend
   # dataset first.

   dsconfig create-placement-algorithm
   --processor-name ou_people_dc_example_dc_com-eb-req-processor
   --algorithm-name example_com_entry_count
   --type entry-counter
   --set enabled:true
   --set "poll-interval:1 m"

   # Note that once the entry-count placement algorithm is created
   # and enabled, we can delete the round-robin algorithm.
   ```

```
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then delete the
# original round-robin algorithm created during the setup
# procedure.

dsconfig delete-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin
```

2. Use `dsconfig` with the --batch-file option to read and execute the commands.

Topology Configuration

Topology configuration enables grouping servers and mirroring configuration changes automatically. It uses a master/slave architecture for mirroring shared data across the topology. All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group. Servers can be added to an existing topology at installation.

Note: To remove a server from the topology, it must be uninstalled with the uninstall tool.

Topology Master Requirements and Selection

A topology master server receives any configuration change from other servers in the topology, verifies the change, then makes the change available to all connected servers. The master always sends a digest of its subtree contents on each update. If the node has a different digest than the master, it knows it's not synchronized. The servers will pull the entire subtree from the master if they detect that they are not synchronized. A server may detect it is not synchronized with the master under the following conditions:

- At the end of its periodic polling interval, if a server's subtree digest differs from that of its master, then it knows it's not synchronized.
- If one or more servers have been added to or removed from the topology, the servers will not be synchronized.

The master of the topology is selected by prioritizing servers by minimum supported product version, most available, newest server version, earliest start time, and startup UUID (a smaller UUID is preferred).

After determining a master, the topology data is reviewed from all available servers (every five seconds by default) to determine if any new information makes a server better suited to being the master. If a new server can be the master, it will communicate that to the other servers, if no other server has advertised that it should be the master. This ensures that all servers accept the same master at approximately the same time (within a few milliseconds of each other). If there is no better master, the initial master maintains the role.

After the best master has been selected for the given interval, the following conditions are confirmed:

- A majority of servers is reachable from that master. (The master server itself is considered while determining this majority.)
- There is only a single master in the entire topology.

If either of these conditions is not met, the topology is without a master and the peer polling frequency is reduced to 100 milliseconds to find a new master as quickly as possible. If there is no master in the topology for more than one minute, a `mirrored-subtree-manager-no-master- found` alarm is raised. If one of the servers in the topology is forced as master with the `force-as-master-for-mirrored-data` option in the Global Configuration object, a `mirrored-subtree-manager-forced-as-master-warning` warning alarm is raised. If multiple servers have been forced as masters, then a `mirrored-subtree-manager-forced-as-master-error` critical alarm will be raised.

Topology Components

When a server is installed, it can be added to an existing topology, which will clone the server's configuration. Topology settings are designed to operate without additional configuration. If required, some settings can be adjusted to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the Global Configuration and in the Config File Handler Backend. Though they are topology settings, they are unique to each server and are not mirrored. Settings must be kept the same on all servers.

The Global Configuration object contains a single topology setting, force-as-master-formirrored- data. This should be set to true on only one of the servers in the topology, and is used only if a situation occurs where the topology cannot determine a master because a majority of servers is not available. A server with this setting enabled will be assigned the role of master, if no suitable master can be determined.

The Config File Handler Backend defines three topology (`mirrored-subtree`) settings:

- `mirrored-subtree-peer-polling-interval` – Specifies the frequency at which the server polls its topology peers to determine if there are any changes that may warrant a new master selection. A lower value will ensure a faster failover, but it will also cause more traffic among the peers. The default value is five seconds. If no suitable master is found, the polling frequency is adjusted to 100 milliseconds until a new master is selected.
- `mirrored-subtree-entry-update-timeout` – Specifies the maximum length of time to wait for an update operation (add, delete, modify or modify-dn) on an entry to be applied by the master on all of the servers in the topology. The default is 10 seconds. In reality, updates can take up to twice as much time as this timeout value if master selection is in progress at the time the update operation was received.
- `mirrored-subtree-search-timeout` – Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

Topology meta-data is stored under the `cn=topology,cn=config` subtree and cluster data is stored under the `cn=cluster,cn=config` subtree. The only setting that can be changed is the cluster name.

Monitor Data for the Topology

Each server has a monitor that exposes that server's view of the topology in its monitor backend, so that peer servers can periodically read this information to determine if there are changes in the topology. Topology data includes the following:

- The server ID of the current master, if the master is not known.
- The instance name of the current master, or if a master is not set, a description stating why a master is not set.
- A flag indicating if this server thinks that it should be the master.
- A flag indicating if this server is the current master.
- A flag indicating if this server was forced as master.
- The total number of configured peers in the topology group.
- The peers connected to this server.
- The current availability of this server.
- A flag indicating whether or not this server is not synchronized with its master, or another node in the topology if the master is unknown.
- The amount of time in milliseconds where multiple masters were detected by this server.
- The amount of time in milliseconds where no suitable server is found to act as master.
- A SHA-256 digest encoded as a base-64 string for the current subtree contents.

The following metrics are included if this server has processed any operations as master:

- The number of operations processed by this server as master.
- The number of operations processed by this server as master that were successful.
- The number of operations processed by this server as master that failed to validate.
- The number of operations processed by this server as master that failed to apply.
- The average amount of time taken (in milliseconds) by this server to process operations as the master.
- The maximum amount of time taken (in milliseconds) by this server to process an operation as the master.

Updating the Server Instance Listener Certificate

To change the SSL certificate for the server, update the keystore and truststore files with the new certificate. The certificate file must have the new certificate in PEM-encoded format, such as:

```
-----BEGIN CERTIFICATE-----

 MIIDKTCCAhGgAwIBAgIEacgGrDANBgkqhkiG9w0BAQsFADBFMR4wHAYDVQQKExVVbmJvdW5kSUQgQ2VydGlmaWN

 GUxIzAhBgNVBAMTGnZtLW1lZGl1bS03My51bmJvdW5kaWQubGFiMB4XDTE1MTAxMjE1MzU0OFoXDTM1MTAwNzE1

 U0OFowRTEeMBwGA1UEChMVVW5ib3VuZElEIENlcnRpZmljYXRlMSMwIQYDVQQDExp2bS1tZWRpdW0tNzMudW5ib

 uZGlkLmxhYjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKN4tAN3o9Yw6Cr9hivwVDxJqF6+aEi9Ir
            GFYLSrggRNXsiAOfWkSMWdIC5vyF5OJ9DlIgvHL4OuqP/
YNEGzKDkgr6MwtUeVSK14+dCixygJGC0nY7k+f0WSCjt
            IHzrmc4WWdrZXmgb
+qv9LupS30JG0FXtcbGkYpjaKXIEqMg4ekz3B5cAvE0SQUFyXEdN4rWOn96nVFkb2CstbiPzA

 gne2tu7paJ6SGFOW0UF7v018XY1m2WHBIoD0WC8nOVLTG9zFUavaOxtlt1TlhClkI4HRMNg8n2EtSTdQRizKuw9
            TXJBb6Kfvnp/
nI73VHRyt47wUVueehEDfLtDP8pMCAwEAAaMhMB8wHQYDVR0OBBYEFMrwjWxl2K+yd9+Y65oKn0g5
            jITgMA0GCSqGSIb3DQEBCwUAA4IBAQBpsBYodblUGew+HewqtO2i8Wt+vAbt31zM5/
kRvo6/+iPEASTvZdCzIBcgl

 etxKGKeCQ0GPeHr42+erakiwmGDlUTYrU3LU5pTGTDLuR2IllTT5xlEhCWJGWipW4q3Pl3cX/9m2ffY/
JLYDfTJao

 JvnXrh7Sg719skkHjWZQgOHXlkPLx5TxFGhAovE1D4qLVRWGohdpWDrIgFh0DVfoyAn1Ws9ICCXdRayajFI4Lc6
            m6SA5+25Y9nno8BhVPf4q5OW6+UDc8MsLbBsxpwvR6RJ5cv3ypfOriTehJsG
+9ZDo7YeqVsTVGwAlW3PiSd9bYP/8
            yu9Cy+0MfcWcSeAE
            -----END CERTIFICATE-----
```

If clients that already have a secure connection established with this server need to be maintained, information about both certificates can reside in the same file (each with their own begin and end headers and footers).

After the keystore and truststore files are updated, run the following dsconfig command to update the server's certificate in the topology registry:

```
$ bin/dsconfig set-server-instance-listener-prop \
  --instance-name server-instance-name \
  --listener-name ldap-listener-mirrored-config \
  --set listener-certificate path-to-new-certificate-file
```

The `listener-certificate` in the topology registry is like a trust store. The public certificates that it has are automatically trusted by the local server. When the local server attempts a secure LDAP connection to a peer, and the peer presents it with its certificate, the local server will check the `listener-certificate` property for that server in the topology registry. If the property contains the peer server's certificate, the local server will trust the peer.

Remove the Self-signed Certificate

The server is installed with a self-signed certificate and key (`ads-certificate`), which are used for internal purposes such as replication authentication, inter-server authentication in the topology registry, reversible password encryption, and encrypted backup or LDIF export. The ads-certificate lives in the keystore file called `ads-truststore` under the server’s `/config` directory. If your deployment requires removing the self-signed certificate, it can be replaced.

The certificate is stored in the topology registry, which enables replacing it on one server and having it mirrored to all other servers in the topology. Any change is automatically mirrored on other servers in the topology. It is stored in human-readable PEM-encoded format and can be updated with dsconfig. The following general steps are required to replace the self-signed certificate:

1. Prepare a new keystore with the replacement key-pair.
2. Update the server configuration to use the new certificate by adding it to the server's list of certificates in the topology registry so that it is trusted by other servers.
3. Update the server's `ads-truststore` file to use the new key-pair.
4. Retire the old certificate by removing it from the topology registry.

Note: Replacing the entire key-pair instead of just the certificate associated with the original private key can make existing backups and LDIF exports invalid. This should be performed immediately after setup or before the key-pair is used. After the first time, only the certificate associated with the private key should have to be changed, for example, to extend its validity period or replace it with a certificate signed by a different CA.

Prepare a New Keystore with the Replacement Key-pair

The self-signed certificate can be replaced with an existing key-pair, or the certificate associated with the original key-pair can be used.

To Use an Existing Key-pair

If a private key and certificate(s) in PEM-encoded format already exist, both the original private key and self-signed certificate can be replaced in ads-truststore with the `manage-certificates` tool.

- The following command imports the keystore file, ads-truststore.new.

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file existing.key \
  --certificate-file existing.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

The certificates listed using the `--certificate-file` options must be ordered so that each subsequent certificate is the issuer for the previous one. So the server certificate comes first, the intermediate certificates next (if any), and the root CA certificate last.

To Use the Certificate Associated with the Original Key-pair

The certificate associated with the original server-generated private key can be replaced with the following commands:

1. Create a CSR for the `ads-certificate`:

```
$ bin/manage-certificates generate-certificate-signing-request \
  --keystore ads-truststore \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --use-existing-key-pair \
  --subject-dn "CN=ldap.example.com,O=Example Corporation,C=US" \
  --output-file ads.csr
```

2. Submit `ads.csr` to a CA for signing.
3. Export the server's private key into `ads.key`:

```
$ bin/manage-certificates export-private-key \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --output-file ads.key
```

4. Import the certificates obtained from the CA (the CA-signed server certificate, any intermediate certificates, and root CA certificate) into `ads-truststore.new`:

   ```
   $ bin/manage-certificates import-certificate \
     --keystore ads-truststore.new \
     --keystore-type JKS \
     --keystore-password-file ads-truststore.pin \
     --alias ads-certificate \
     --private-key-file ads.key \
     --certificate-file new-ads.crt \
     --certificate-file intermediate.crt \
     --certificate-file root-ca.crt
   ```

To Update the Server Configuration to Use the New Certificate

To update the server to use the desired key-pair, the `inter-server-certificate` property for the server instance must first be updated in the topology registry. The old and the new certificates may appear within their own begin and end headers in the `inter-server-certificate` property to support transitioning from the old certificate to the new one.

1. Export the server's old ads-certificate into `old-ads.crt`::

   ```
   $ bin/manage-certificates export-certificate \
     --keystore ads-truststore \
     --keystore-password-file ads-truststore.pin \
     --alias ads-certificate \
     --export-certificate-chain \
     --output-file old-ads.crt
   ```
2. Concatenate the old, new certificate, and issuer certificates into one file. On Windows, an editor like notepad can be used. On Unix platforms, use the following command:

   ```
   $ cat old-ads.crt new-ads.crt intermediate.crt root-ca.crt > chain.crt
   ```
3. Update the `inter-server-certificate` property for the server instance in the topology registry using `dsconfig`:

   ```
   $ bin/dsconfig -n set-server-instance-prop \
     --instance-name instance-name \
     --set “inter-server-certificate<chain.crt”
   ```

To Update the ads-truststore File to Use the New Key-pair

The server will still use the old `ads-certificate`. When the new `ads-certificate` needs to go into effect, the old `ads-truststore` file must be replaced with `ads-truststore.new` in the server's `config` directory.

- Move the file.

  ```
  $ mv ads-truststore.new ads-truststore
  ```

To Retire the Old Certificate

The old certificate is retired by removing it from the topology registry when it has expired. All existing encrypted backups and LDIF exports are not affected because the public key in the old and new server certificates are the same, and the private key will be able to decrypt them.

- Perform the following commands:

  ```
  $ cat new-ads.crt intermediate.crt root-ca.crt<chain.crt
  ```

  ```
  $ bin/dsconfig -n set-server-instance-prop \
    --instance-name instance-name \
    --set “inter-server-certificate<chain.crt”
  ```

Using the Configuration API

PingData servers provide a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers should allow the application/json content type.

The server includes a servlet extension that provides read and write access to the server's configuration over HTTP. The extension is enabled by default for new installations, and can be enabled for existing deployments by simply adding the extension to one of the server's HTTP Connection Handlers, as follows:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add http-servlet-extension:Configuration
```

The API is made available on the HTTPS Connection handler's host:port in the `/config` context. Due to the potentially sensitive nature of the server's configuration, the HTTPS Connection Handler should be used for hosting the Configuration extension.

Authentication and Authorization with the Configuration API

Clients must use HTTP Basic authentication to authenticate to the Configuration API. If the username value is not a DN, then it will be resolved to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper, or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. For example:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Configuration \
  --set "identity-mapper:Alternative Identity Mapper”
```

To access configuration information, users must have the appropriate privileges:

- To access the `cn=config` backend, users must have the `bypass-acl` privilege or be allowed access to the configuration using an ACI.
- To read configuration information, users must have the `config-read` privilege.
- To update the configuration, users must have the `config-write` privilege.

Relationship Between the Configuration API and the dsconfig Tool

The Configuration API is designed to mirror the `dsconfig` tool, using the same names for properties and object types. Property names are presented as hyphen case in `dsconfig` and as camel-case attributes in the API. In API requests that specify property names, case is not important. Therefore, `baseDN` is the same as `baseDn`. Object types are represented in hyphen case. API paths mirror what is in `dsconfig`. For example, the `dsconfig list-connection-handlers` command is analogous to the API's `/config/connection-handlers` path. Object types that appear in the schema URNs adhere to a `type:subtype` syntax. For example, a Local DB Backend's schema URN is `urn:unboundid:schemas:configuration:2.0:backend:local-db`. Like the `dsconfig` tool, all configuration updates made through the API are recorded in `logs/config-audit.log`.

The API includes the filter, sort, and pagination query parameters described by the SCIM specification. Specific attributes may be requested using the attributes query parameter, whose value must be a comma-delimited list of properties to be returned, for example `attributes=baseDN,description`. Likewise, attributes may be excluded from responses by specifying the `excludedAttributes` parameter.

Operations supported by the API are those typically found in REST APIs:

HTTP Method	Description	Related dsconfig Example
GET	Lists the properties of an object when used with a path representing an object, such as `/config/global-configuration` or `/config/backends/userRoot`. Can also list objects when used with a path representing a parent relation, such as `/config/backends`.	`get-backend-prop`, `list-backends`, `get-global-configuration-prop`
POST	Creates a new instance of an object when used with a relation parent path, such as `/config/backends`.	`create-backend`
PUT	Replaces the existing properties of an object. A PUT operation is similar to a PATCH operation, except that the PATCH identifies the difference between an existing target object and a supplied source object. Only those properties in the source object are modified in the target object. The target object is specified using a path, such as `/config/backends/userRoot`.	`set-backend-prop`, `set-global-configuration-prop`
PATCH	Updates the properties of an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`set-backend-prop`, `set-global-configuration-prop`
DELETE	Deletes an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`delete-backend`

The OPTIONS method can also be used to determine the operations permitted for a particular path.

Object names, such as `userRoot` in the Description column, must be URL-encoded for use in the path segment of a URL. For example, `%20` must be used in place of spaces, and `%25` is used in place of the percent (%) character. The URL for accessing the HTTP Connection Handler object is:

```
/config/connection-handlers/http%20connection%20handler
```

GET Example

The following is a sample GET request for information about the `userRoot` backend:

```
GET /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
```

The response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
   "id": "userRoot",
   "meta": {
     "resourceType": "Local DB Backend",
     "location": "http://localhost:5033/config/backends/userRoot"
   },
   "backendID": "userRoot2",
   "backgroundPrime": "false",
   "backupFilePermissions": "700",
   "baseDN": [
     "dc=example2,dc=com"
   ],
   "checkpointOnCloseCount": "2",
   "cleanerThreadWaitTime": "120000",
   "compressEntries": "false",
   "continuePrimeAfterCacheFull": "false",
   "dbBackgroundSyncInterval": "1 s",
   "dbCachePercent": "10",
   "dbCacheSize": "0 b",
   "dbCheckpointerBytesInterval": "20 mb",
   "dbCheckpointerHighPriority": "false",
   "dbCheckpointerWakeupInterval": "1 m",
   "dbCleanOnExplicitGC": "false",
   "dbCleanerMinUtilization": "75",
   "dbCompactKeyPrefixes": "true",
   "dbDirectory": "db",
   "dbDirectoryPermissions": "700",
   "dbEvictorCriticalPercentage": "0",
   "dbEvictorLruOnly": "false",
   "dbEvictorNodesPerScan": "10",
   "dbFileCacheSize": "1000",
   "dbImportCachePercent": "60",
   "dbLogFileMax": "50 mb",
   "dbLoggingFileHandlerOn": "true",
   "dbLoggingLevel": "CONFIG",
   "dbNumCleanerThreads": "0",
   "dbNumLockTables": "0",
   "dbRunCleaner": "true",
   "dbTxnNoSync": "false",
   "dbTxnWriteNoSync": "true",
   "dbUseThreadLocalHandles": "true",
   "deadlockRetryLimit": "10",
   "defaultCacheMode": "cache-keys-and-values",
   "defaultTxnMaxLockTimeout": "10 s",
   "defaultTxnMinLockTimeout": "10 s",
   "enabled": "false",
   "explodedIndexEntryThreshold": "4000",
   "exportThreadCount": "0",
   "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
   "externalTxnDefaultMaxLockTimeout": "100 ms",
   "externalTxnDefaultMinLockTimeout": "100 ms",
   "externalTxnDefaultRetryAttempts": "2",
   "hashEntries": "false",
   "id2childrenIndexEntryLimit": "66",
   "importTempDirectory": "import-tmp",
   "importThreadCount": "16",
   "indexEntryLimit": "4000",
   "isPrivateBackend": "false",
   "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
   "jeProperty": [
     "je.cleaner.adjustUtilization=false",
     "je.nodeMaxEntries=32"
   ],
   "numRecentChanges": "50000",
   "offlineProcessDatabaseOpenTimeout": "1 h",
   "primeAllIndexes": "true",
   "primeMethod": [
     "none"
   ],
   "primeThreadCount": "2",
   "primeTimeLimit": "0 ms",
   "processFiltersWithUndefinedAttributeTypes": "false",
   "returnUnavailableForUntrustedIndex": "true",
   "returnUnavailableWhenDisabled": "true",
   "setDegradedAlertForUntrustedIndex": "true",
   "setDegradedAlertWhenDisabled": "true",
   "subtreeDeleteBatchSize": "5000",
   "subtreeDeleteSizeLimit": "5000",
   "uncachedId2entryCacheMode": "cache-keys-only",
   "writabilityMode": "enabled"
}
```

GET List Example

The following is a sample GET request for all local backends:

```
GET /config/backends/
Host: example.com:5033
Accept: application/scim+json
```

The response (which has been shortened):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=topology,cn=config"
      ],
      "enabled": "true",
      "isPrivateBackend": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.LDIFBackend",
      "ldifFile": "config/admin-backend.ldif",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "false",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
      ],
      "id": "ads-truststore",
      "meta": {
        "resourceType": "Trust Store Backend",
        "location": "http://localhost:5033/config/backends/ads-
truststore"
      },
      "backendID": "ads-truststore",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=ads-truststore"
      ],
      "enabled": "true",
      "javaClass":
"com.unboundid.directory.server.backends.TrustStoreBackend",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "true",
      "trustStoreFile": "config/server.keystore",
      "trustStorePin": "********",
      "trustStoreType": "JKS",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:alarm"
      ],
      "id": "alarms",
      "meta": {
        "resourceType": "Alarm Backend",
        "location": "http://localhost:5033/config/backends/alarms"
      },
  ...
```

PATCH Example

Configuration can be modified using the HTTP PATCH method. The PATCH request body is a JSON object formatted according to the SCIM patch request. The Configuration API, supports a subset of possible values for the path attribute, used to indicate the configuration attribute to modify.

The configuration object's attributes can be modified in the following ways. These operations are analogous to the `dsconfig modify-[object]` options.

- An operation to set the single-valued `description` attribute to a new value:

```
{
  "op" : "replace",
  "path" : "description",
  "value" : "A new backend."
}
```

is analogous to:

```
$ dsconfig set-backend-prop
  --backend-name userRoot \
  --set "description:A new backend"
```

- An operation to add a new value to the multi-valued `jeProperty` attribute:

```
{
  "op" : "add",
  "path" : "jeProperty",
  "value" : "je.env.backgroundReadLimit=0"
}
```

is analogous to:

```
$ dsconfig  set-backend-prop --backend-name userRoot \
  --add je-property:je.env.backgroundReadLimit=0
```

- An operation to remove a value from a multi-valued property. In this case, path specifies a SCIM filter identifying the value to remove:

```
{
  "op" : "remove",
  "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --remove je-property:je.cleaner.adjustUtilization=false
```

- A second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify, and a SCIM filter whose attribute is value:

  ```
  {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --remove je-property:je.nodeMaxEntries=32
  ```

- An option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value:

  ```
  {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --reset id2childrenIndexEntryLimit
  ```

The following is the full example request. The API responds with the entire modified configuration object, which may include a SCIM extension attribute `urn:unboundid:schemas:configuration:messages` containing additional instructions:

```
PATCH /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json

{
  "schemas" : [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  } ]
}
```

Example response:

```
{
 "schemas": [
```

```
  "urn:unboundid:schemas:configuration:2.0:backend:local-db"
],
"id": "userRoot2",
"meta": {
  "resourceType": "Local DB Backend",
  "location": "http://example.com:5033/config/backends/userRoot2"
},
"backendID": "userRoot2",
"backgroundPrime": "false",
"backupFilePermissions": "700",
"baseDN": [
  "dc=example2,dc=com"
],
"checkpointOnCloseCount": "2",
"cleanerThreadWaitTime": "120000",
"compressEntries": "false",
"continuePrimeAfterCacheFull": "false",
"dbBackgroundSyncInterval": "1 s",
"dbCachePercent": "10",
"dbCacheSize": "0 b",
"dbCheckpointerBytesInterval": "20 mb",
"dbCheckpointerHighPriority": "false",
"dbCheckpointerWakeupInterval": "1 m",
"dbCleanOnExplicitGC": "false",
"dbCleanerMinUtilization": "75",
"dbCompactKeyPrefixes": "true",
"dbDirectory": "db",
"dbDirectoryPermissions": "700",
"dbEvictorCriticalPercentage": "0",
"dbEvictorLruOnly": "false",
"dbEvictorNodesPerScan": "10",
"dbFileCacheSize": "1000",
"dbImportCachePercent": "60",
"dbLogFileMax": "50 mb",
"dbLoggingFileHandlerOn": "true",
"dbLoggingLevel": "CONFIG",
"dbNumCleanerThreads": "0",
"dbNumLockTables": "0",
"dbRunCleaner": "true",
"dbTxnNoSync": "false",
"dbTxnWriteNoSync": "true",
"dbUseThreadLocalHandles": "true",
"deadlockRetryLimit": "10",
"defaultCacheMode": "cache-keys-and-values",
"defaultTxnMaxLockTimeout": "10 s",
"defaultTxnMinLockTimeout": "10 s",
"description": "123",  "enabled": "false",
"explodedIndexEntryThreshold": "4000",
"exportThreadCount": "0",
"externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
"externalTxnDefaultMaxLockTimeout": "100 ms",
"externalTxnDefaultMinLockTimeout": "100 ms",
"externalTxnDefaultRetryAttempts": "2",
"hashEntries": "false",
"importTempDirectory": "import-tmp",
"importThreadCount": "16",
"indexEntryLimit": "4000",
"isPrivateBackend": "false",
"javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
"jeProperty": [  "\"je.env.backgroundReadLimit=0\""
],
"numRecentChanges": "50000",
"offlineProcessDatabaseOpenTimeout": "1 h",
"primeAllIndexes": "true",
 "primeMethod": [
   "none"
 ],
 "primeThreadCount": "2",
 "primeTimeLimit": "0 ms",
 "processFiltersWithUndefinedAttributeTypes": "false",
 "returnUnavailableForUntrustedIndex": "true",
 "returnUnavailableWhenDisabled": "true",
 "setDegradedAlertForUntrustedIndex": "true",
 "setDegradedAlertWhenDisabled": "true",
 "subtreeDeleteBatchSize": "5000",
 "subtreeDeleteSizeLimit": "5000",
 "uncachedId2entryCacheMode": "cache-keys-only",
 "writabilityMode": "enabled",
 "urn:unboundid:schemas:configuration:messages:2.0": {
   "requiredActions": [
     {
       "property": "jeProperty",
       "type": "componentRestart",
       "synopsis": "In order for this modification to take effect,
          the component must be restarted, either by disabling and
          re-enabling it, or by restarting the server"
     },
     {
       "property": "id2childrenIndexEntryLimit",
       "type": "other",
       "synopsis": "If this limit is increased, then the contents
          of the backend must be exported to LDIF and re-imported to
          allow the new limit to be used for any id2children keys
          that had already hit the previous limit."
     }
   ]
 }
}
```

Configuration API Paths

The Configuration API is available under the /config path. A full listing of supported sub-paths is available by accessing the base `/config/ResourceTypes` endpoint:

```
GET /config/ResourceTypes
Host: example.com:5033
Accept: application/scim+json
```

Sample response (abbreviated):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "dsee-compat-access-control-handler",
      "name": "DSEE Compat Access Control Handler",
      "description": "The DSEE Compat Access Control
              Handler provides an implementation that uses syntax
              compatible with the Sun Java System Directory Server
              Enterprise Edition access control handler.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/dsee-compat-
access-control-handler"
      }
    },
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "access-control-handler",
      "name": "Access Control Handler",
      "description": "Access Control Handlers manage the
              application-wide access control. The server's access
              control handler is defined through an extensible
              interface, so that alternate implementations can be created.
              Only one access control handler may be active in the server
                        at any given time.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/access-
control-handler"
      }
    },
    {
 ...
```

The response's `endpoint` elements enumerate all available sub-paths. The path `/config/access-control-handler` in the example can be used to get a list of existing access control handlers, and create new ones. A path containing an object name such as `/config/backends/{backendName}`, where `{backendName}` corresponds to an existing backend (such as `userRoot`) can be used to obtain an object's properties, update the properties, or delete the object.

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted nor created. These paths can be differentiated from others by their singular, rather than plural, relation name (for example `global-configuration`).

Sorting and Filtering Objects

The Configuration API supports SCIM parameters for filter, sorting, and pagination. Search operations can specify a SCIM filter used to narrow the number of elements returned. See the SCIM specification for the full set of operations for SCIM filters. Clients can also specify sort parameters, or paging parameters. Include or exclude attributes can be specified in both get and list operations.

GET Parameter	Description
filter	Values can be simple SCIM filters such as `id eq "userRoot"` or compound filters like `meta.resourceType eq "Local DB Backend"` and `baseDn co "dc=exmple,dc=com"`.
sortBy	Specifies a property value by which to sort.
sortOrder	Specifies either `ascending` or `descending` alphabetical order.
startIndex	1-based index of the first result to return.
count	Indicates the number of results per page.

Updating Properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH. With PUT, the server computes the differences between the object in the request with the current version in the server, and performs modifications where necessary. The server will never remove attributes that are not specified in the request. The API responds with the entire modified object.

Request:

```
PUT /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
{
  "description" : "A new description."
}
```

Response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode":
  "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "abc",
  "enabled": "true",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior":
  "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "true",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "numRecentChanges": "50000",  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative Actions

Updating a property may require an administrative action before the change can take effect. If so, the server will return `200 Success`, and any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

For example, changing the `jeProperty` of a backend will result in the following:

```
"urn:unboundid:schemas:configuration:messages:2.0": {
    "required-actions": [
      {
       "property": "baseContextPath",
       "type": ""componentRestart",
       "synopsis": "In order for this modification to take effect, the
 component
                 must be restarted, either by disabling and re-enabling it,
 or
                 by restarting the server"
      },
      {
        "property": {
        "type": "other",
        "synopsis": "If this limit is increased, then the
                 contents of the backend must be exported to LDIF
                 and re-imported to allow the new limit to be used
                 for any id2children keys that had already hit the
                 previous limit."
      }
    ]
  }
```

Updating Servers and Server Groups

Servers can be configured as part of a server group, so that configuration changes that are applied to a single server, are then applied to all servers in a group. When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query attribute. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `single-server` or `server-group` can be specified. For example:

```
http://localhost:8082/config/backends/userRoot?applyChangeTo=single-server
```

Configuration API Responses

Clients of the API should examine the HTTP response code in order to determine the success or failure of a request. The following are response codes and their meanings:

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, or object properties, administrative actions.
204 No Content	The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation.	None.
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message.
401 Unauthorized	User authentication is required. Some user agents such as browsers may respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid.	None.
403 Forbidden	The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted.	None.
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message.
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists, or an attempt was made to delete an object that is referred to by another object.	Error summary and optional message.

Response Code	Description	Response Body
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None.
500 Server Error	The server encountered an unexpected error. Please report server errors to customer support.	Error summary and optional message.

An application that uses the Configuration API should limit dependencies on particular text appearing in error message content. These messages may change, and their presence may depend on server configuration. Use the HTTP return code and the context of the request to create a client error message. The following is an example encoded error message:

```
{
 "schemas": [
   "urn:ietf:params:scim:api:messages:2.0:Error"
 ],
 "status": 404,
 "scimType": null,
 "detail": "The Local DB Index does not exist."
}
```

Generating a Summary of Configuration Components

The Directory Proxy Server provides a `config-diff` tool that generates a summary of the configuration in a local or remote directory server instance. The tool is useful when comparing configuration settings on the directory server instance when troubleshooting issues or when verifying configuration settings on newly-added servers to your network. The tool can interact with the local configuration regardless of whether the server is running or not.

Run the `config-diff` --help option to view other available tool options.

To Generate a Summary of Configuration Components

- Run the `config-diff` tool to generate a summary of the configuration components on the directory server instance. The following command runs a summary on a local online server.

  ```
  $ bin/config-diff
  ```
- The following example compares the current configuration of the local server to the baseline, pre-installation configuration, ignoring any changes that could be made by the installer, and writes the output to the `configuration-steps.dsconfig` file. This provides a script that can be used to configure a newly installed server identically to the local server:

  ```
  $ bin/config-diff --sourceLocal \
    --sourceBaseline \
    --targetLocal \
    --exclude differs-after-install \
    --outputFile configuration-steps.dsconfig
  ```

Configuring Server Groups

The PingDirectoryProxy Server provides a mechanism for setting up administrative domains that synchronize configuration changes among servers in a server group. After you have set up a server group, you can make an update on one server using `dsconfig`, then you can apply the change to the other servers in the group using the --applyChangeTo server-group option of the `dsconfig` non-interactive command. If you want to apply the change to one server in the group, use the --applyChangeTo single-server option. When using `dsconfig` in interactive command-line mode, you will be asked if you want to apply the change to a single server or to all servers in the server group.

About the Server Group Example

You can create an administrative server group using the `dsconfig` tool. The general process is to create a group, add servers to the group, and then set a global configuration property to use the server group. If you are configuring a replication topology, then you must configure the replicas to be in a server group as outlined in Replication Configuration.

The following example procedure adds three Directory Proxy Server instances into the server group labelled "group-one".

To Create a Server Group

1. Create a group called "group-one" using `dsconfig`.

   ```
   $ bin/dsconfig create-server-group --group-name group-one
   ```

2. Add any directory server to the server group. If you have set up replication between a set of servers, these server entries will have already been created by the `dsreplication` enable command.

   ```
   $ bin/dsconfig set-server-group-prop --group-name group-one --set
    member:server1

   $ bin/dsconfig set-server-group-prop --group-name group-one --set
    member:server2

   $ bin/dsconfig set-server-group-prop --group-name group-one --set
    member:server3
   ```

3. Set a global configuration property for each of the servers that should share changes in this group.

   ```
   $ bin/dsconfig set-global-configuration-prop \
     --set configuration-server-group:group-one
   ```

4. Test the server group. In this example, enable the log publisher for each directory server in the group, server-group, by using the --applyChangeTo server-group option.

   ```
   $ bin/dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Audit Logger" \
     --set enabled:true \
     --applyChangeTo server-group
   ```

5. View the property on the first directory server instance.

   ```
   $ bin/dsconfig get-log-publisher-prop \
     --publisher-name "File-Based Audit Logger" \
     --property enabled
   ```

   ```
   Property : Value(s)
   ---------:---------
   enabled : true
   ```

6. Repeat step 5 on the second and third directory server instance.

7. Test the server group by disabling the log publisher on the first directory server instance by using the --applyChangeTo single-server.

   ```
   $ bin/dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Audit Logger" \
     --set enabled:disabled \
     --applyChangeTo single-server
   ```

8. View the property on the first directory server instance. The first directory server instance should be disabled.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : false
```

9. View the property on the second directory server instance. Repeat this step on the third directory server instance to verify that the property is still enabled on that server.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
---------:---------
enabled : true
```

Domain Name Service (DNS) Caching

If needed, two global configuration properties can be used to control the caching of hostname-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system. Use the `dsconfig` tool to configure these properties.

- **network-address-cache-ttl** – Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a hostname-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.
- **network-address-outage-cache-enabled** – Caches hostname-to-IP address results in the event of a DNS outage. This is set to true by default, meaning name resolution results are cached. Unexpected service interruptions may occur during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

IP Address Reverse Name Lookups

PingData servers do not explicitly perform numeric IP address-to-hostname lookups. However, address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing may trigger implicit reverse name lookups. For more information about how address masks are configured in the server, review the following information for each server:

- ACI dns: bind rules under *Managing Access Control* (Directory Server and Directory Proxy Server)
- `ds-auth-allowed-address`: *Adding Operational Attributes that Restrict Authentication* (Directory Server)
- Connection Criteria: *Restricting Server Access Based on Client IP Address* (Directory Server and Directory Proxy Server)
- Connection Handlers: restrict server access using Connection Handlers (Configuration Reference Guide for all servers)

Configuring Traffic Through a Load Balancer

If a PingData server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, then it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request. If the actual client's IP address must be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and PingData server. For PingData servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP), and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

On the load balancer, configure settings to provide the `X-Forwarded-*` information, such as `X-Forwarded-Host:`. See the product documentation for the device type.

Managing Root Users Accounts

The PingDirectoryProxy Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file (for example, under `cn=Root DNs,cn=config`). The root user is the LDAP-equivalent of a UNIX super-user account and inherits its read-write privileges from the default root privilege set. Root users can be created and updated with the `dsconfig` tool. Root user entries are stored in the server’s configuration.The following is a sample command to create a new root user:

```
bin/dsconfig create-root-dn-user --user-name "Joanne Smith" \
  --set last-name:Smith \
  --set first-name:Joanne \
  --set user-id:jsmith \
  --set 'email-address:jsmith@example.com' \
  --set mobile-telephone-number:8889997777 \
  --set home-telephone-number:5556667777 \
  --set work-telephone-number:4445556666
```

To limit full access to all of the Directory Proxy Server, create separate administrator accounts with limited privileges so that you can identify the administrator responsible for a particular change. Having separate user accounts for each administrator also makes it possible to enable password policy functionality (such as password expiration, password history, and requiring secure authentication) for each administrator.

Default Root Privileges

The PingDirectoryProxy Server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments.

Note: Creating restricted root user accounts requires assigning privileges and necessary access controls for actions on specific data or backends. Access controls are determined by how the directory is configured and the structure of your data. See Chapter 16: Managing Access Controls for more information.

The following set of root privileges are available to each root user DN:

Table 2: Default Root Privileges

Privilege	Description
audit-data-security	Allows the associated user to execute data security auditing tasks.
backend-backup	Allows the user to perform backend backup operations.
backend-restore	Allows the user to perform backend restore operations.
bypass-acl	Allows the user to bypass access control evaluation.
config-read	Allows the user to read the server configuration.
config-write	Allows the user to update the server configuration.
disconnect-client	Allows the user to terminate arbitrary client connections.

Privilege	Description
ldif-export	Allows the user to perform LDIF export operations.
ldif-import	Allows the user to perform LDIF import operations.
lockdown-mode	Allows the user to request a server lockdown.
manage-topology	Allows the user to modify topology setting.
metrics-read	Allows the user to read server metrics.
modify-acl	Allows the user to modify access control rules.
password-reset	Allows the user to reset user passwords but not their own. The user must also have privileges granted by access control to write the user password to the target entry.
permit-get-password-policy-state-issues	Allows the user to access password policy state issues.
privilege-change	Allows the user to change the set of privileges for a specific user, or to change the set of privileges automatically assigned to a root user.
server-restart	Allows the user to request a server restart.
server-shutdown	Allows the user to request a server shutdown.
soft-delete-read	Allows the user access to soft-deleted entries.
stream-values	Allows the user to perform a stream values extended operation that obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
third-party-task	Allows the associated user to invoke tasks created by third-party developers.
unindexed-search	Allows the user to perform an unindexed search in the Oracle Berkeley DB Java Edition backend.
update-schema	Allows the user to update the server schema.
use-admin-session	Allows the associated user to use an administrative session to request that operations be processed using a dedicated pool of worker threads.

The Directory Proxy Server provides other privileges that are not assigned to the root user DN by default but can be added using the `ldapmodify` tool (see Modifying Individual Root User Privileges) for more information.

Table 3: Other Available Privileges

Privilege	Description
bypass-pw-policy	Allows the associated user bypass password policy rules and restrictions.
bypass-read-aci	Allows the associated user to bypass access control checks performed by the server for bind, compare, and search operations. Access control evaluation may still be enforced for other types of operations.
jmx-notify	Allows the associated user to subscribe to receive JMX notifications.
jmx-read	Allows the associated user to perform JMX read operations.
jmx-write	Allows the associated user to perform JMX write operations.
permit-externally-processed-authentication	Allows the associated user accept externally processed authentication.
permit-proxied-mschapv2-details	Allows the associated user to permit MS-CHAP V2 handshake protocol.

Privilege	Description
proxied-auth	Allows the associated user to accept proxied authorization.

Configuring Locations

PingDirectoryProxy Server defines locations, both for the LDAP external servers and the proxy server instances themselves. A location defines a collection of servers that share access and latency characteristics. For example, your deployment might include two data centers, one in the east and one in the west. These data centers would be configured as two locations in the Directory Proxy Server. Each location is associated with a name and an ordered list of failover locations, which could be used if none of the servers in the preferred location are available. You can define these locations using the Administrative Console or the command line.

The Directory Proxy Server itself is also associated with a location. This location is specified in the global configuration properties of the Directory Proxy Server. If the load balancing algorithm's `use-location` property is set to true, then the load balancing component of the Directory Proxy Server refers to the Directory Proxy Server's location to determine the external servers it prefers to communicate with.

To Configure Locations Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server main menu, enter the number corresponding to location configuration. Then, enter the number to create a new location.
7. Enter the name of the new location. This example demonstrates configuring a location called East. Enter **f** to finish configuring the location. Repeat this procedure to create a location called West.

```
>>>>  Enter a name for the location that you want to create: east

>>>>  Configure the properties of the location

      Property                   Value(s)
      -----------------------------------
   1) description                  -
   2) preferred-failover-location  -

   ?) help
   f) finish - create the new location
   d) display the equivalent dsconfig arguments to
      create this object
   b) back
   q) quit

Enter choice [b]: f
```

8. Next, edit the configuration of an existing location, in this example a location named East.

```
>>>> Location menu
What would you like to do?

  1) List existing locations
  2) Create a new location
  3) View and edit an existing location
  4) Delete an existing location

  b) back
  q) quit

Enter choice [b]: 3

>>>> Select the location from the following list:
   1) East
   2) West

   b) back
   q) quit

Enter choice [b]: 1
```

9. Define the preferred failover location property for East. This property provides alternate locations that can be used if servers in this location are not available. If more than one location is provided, the Directory Proxy Server tries the locations in the order listed.

```
 >>>> Configure the properties of the Location

     Property                   Value(s)
     -----------------------------------
  1) description                  -
  2) preferred-failover-location  -

  ?) help
  f) finish - create the new location
  d) display the equivalent dsconfig arguments to create this object
  b) back
  q) quit

Enter choice [b]: 2

...

Do you want to modify the 'preferred-failover-location' property?

   1) Add one or more values

   ?) help
   q) quit

Enter choice [1]: 2

Select the locations you wish to add:

  1) East
  2) West
  3) Create a new location
  4) Add all locations

...

Enter one or more choices separated by commas[b]: 2
```

10. Verify and apply your change to the property.

```
Do you want to modify the 'preferred-failover-location' property?

  1) Use the value: West
  2) Add one or more values
  3) Remove one or more values
  4) Leave undefined
  5) Revert changes

  ?) help
  q) quit

Enter choice [1]:

>>>> Configure the properties of the location

     Property                   Value(s)
     -----------------------------------
  1) description                  -
  2) preferred-failover-location  West

  ?) help
  f) finish - apply any changes to the Location
  d) display the equivalent dsconfig command lines to either
     re-create this object or only to apply pending changes
  b) back
  q) quit

Enter choice [b]: f
```

11. Repeat steps 8 and 9 for the West location, assigning it a failover location of East.

To Modify Locations Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server main menu, enter the number corresponding to Global Configuration. Then enter the number to view and edit the Global Configuration.
7. Enter the number associated with the location configuration property.

```
Enter choice [b]: 2
```

8. Specify a new location for this Directory Proxy Server instance, in this example the East location. Operations involving communications with other servers may prefer servers in the same location to ensure low-latency responses.

```
>>>> Configuring the 'location' property

...

Do you want to modify the 'location' property?

  1) Leave undefined
  2) Change it to the location: East
  3) Change it to the location: West
  4) Create a new location

  b) back
  q) quit

Enter choice [b]: 2
```

9. Enter **f** to finish the operation.

```
Enter choice [b]: f
```

Configuring Batched Transactions

You can configure the Directory Proxy Server to use batched transactions in both simple and entry-balanced configurations. The batched transactions feature supports two implementations: the standard LDAP transactions per RFC 5805 and the PingData proprietary implementation, known as the *multi-update extended operation*. Batched transactions can be used through the Directory Proxy Server in both simple and entry-balanced configurations although only in cases in which all operations within the transaction request may be processed within the same backend server and within the same Berkeley DB JE backend. Batched transactions cannot be processed across multiple servers or multiple Directory Server backends.

The multi-update extended operation makes it possible to submit multiple updates in a single request. These updates may be processed either as individual operations or as a single atomic unit. When the Directory Proxy Server receives a Start Batched Transaction request, it will queue all associated operations in memory until the End Batched Transaction request is received with the intention to commit, at which point the set of operations is sent as a single multi-update extended request to the Directory Server.

Add, delete, modify, modify DN, and password modify extended operations may be included in the set of operations processed during a batch transaction. The operations are processed sequentially in the order in which they were included in the extended request. If an error occurs while processing an operation in the set, then the server can be instructed to continue the processing or to cancel any remaining operations. If the operations are not cancelled, you can configure the server to process all operations as a single atomic unit.

Because of this use of multi-update, the external Directory Server must be configured to allow multi-update extended requests made by the Directory Proxy Server on behalf of the DN submitting the batched transaction. For example, the following Directory Server `dsconfig` command grants anonymous access to the multi-update extended request. The submitter of the request still needs access rights for the individual operations within the multiple-update.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(extop="1.3.6.1.4.1.30221.2.6.17")(version 3.0;
 acl "Anonymous access to multi-update extended request"; allow (read)
 userdn="ldap:///anyone";)'
```

To Configure Batched Transactions

Batched transactions are managed by the Batched Transactions Extended Operation Handler. You can use it to configure the start transaction and end transaction operations used to indicate the set of add, delete, modify, modify DN, and/or password modify operations as a single atomic unit.

1. You can configure batched transactions using the `dsconfig` command as follows:

```
$ bin/dsconfig set-extended-operation-handler-prop \
  --handler-name "Batched Transactions" \
  --set enabled:true
```

2. Configure the external servers to allow the multi-update extended operation by granting access rights to the feature. See example in the previous section.

Configuring Server Health Checks

You can use the PingDirectoryProxy Server to configure different types of health checks for your deployment. The health checks define external server availability as either being available, unavailable, or degraded. The external server health is given a value from 0 to 10, which is used to determine if the server is available and how that server compares to other servers with the same state. Load-balancing algorithms can be used to check the score and prefer servers with higher scores over those with lower scores.

An individual health check can be defined for use against all external servers or assigned to individual external servers, as determined by the `use-for-all-servers` parameter within the health check configuration object. If `use-for-all-servers` is set to true, the Directory Proxy Server applies the health check to all external servers in all locations. If `use-for-all-servers` is set to false, then the health check is only employed against an external server if the configuration object for that external server lists the health check.

For more information about health checks and the type of health checks supported by PingDirectoryProxy Server, see About LDAP Health Checks.

About the Default Health Checks

By default, the Directory Proxy Server has two health check instances enabled for use on all servers:

- **Consume Admin Alerts**. This health check detects administrative alerts from the Directory Server, as soon as they are issued, by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the Directory Server. When the Directory Proxy Server is notified by the Directory Server of a new alert, it immediately retrieves the base `cn=monitor` entry of the Directory Server. If this entry has a value for the `unavailable-alert-type` attribute, then the Directory Proxy Server will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the Directory Proxy Server will consider it degraded.
- **Get Root DSE**. This health check detects if the root DSE entry exists on the LDAP external server. As this entry always exists on an PingDirectory Server, the absence of the entry suggests that the LDAP external server may be degraded or unavailable.

About Creating a Custom Health Check

You can create a new health check from scratch or use an existing health check as a template for the configuration of a new health check. If you choose to create a custom health check, you can create one of the following types:

- **Admin Alert Health Check**. This health check watches for administrative alerts generated by the LDAP external server to determine whether the server has entered a degraded or unavailable state.

- **Groovy Scripted LDAP Health Check**. This health check allows you to create custom LDAP health checks in a dynamically-loaded Groovy script, which implements the `ScriptedLDAPHealthCheck` class defined in the Server SDK.
- **Replication Backlog Health Check**. While the Admin Alert Health Check consumes replication backlog alerts emitted from external servers, a finer definition of external server health based on replication backlog can be defined with this health check. If a server falls too far behind in replication, then the Directory Proxy Server can stop sending requests to it. A server is classified as degraded or unavailable if the threshold is reached for the number of backlogged changes, the age of the oldest backlogged change, or both.
- **Search LDAP Health Check**. This health check performs searches on an LDAP external server and gauges the health of the server depending if the expected results were returned within an acceptable response time. For example, if an error occurs while attempting to communicate with the server, then the server is considered unavailable. You can also apply filters to the results to use values within the monitor entry as indicators of server health.
- **Third Party LDAP Health Check**. This health check allows you to define LDAP health check implementations in third-party code using the Server SDK.
- **Work Queue Busyness Health Check**. This health check may be used to monitor the percentage of time that worker threads in backend servers spend processing requests.

To Configure a Health Check Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server main menu, enter the number corresponding to LDAP health checks. Enter the number to create a new LDAP Health Check, then press `n` to create a new health check from scratch.
7. Select the type of health check you want to create. This example demonstrates the creation of a new search LDAP health check.

```
>>> Select the type of LDAP Health Check that you want to create:

      1)  Admin Alert LDAP Health Check
      2)  Custom LDAP Health Check
      3)  Groovy Scripted LDAP Health Check
      4)  Replication Backlog LDAP Health Check
      5)  Search LDAP Health Check
      6)  Third Party LDAP Health Check
      7)  Work Queue Busyness LDAP Health Check

      ?) help
      c) cancel
      q) quit

Enter choice [c]: 5
```

8. Specify a name for the new health check. In this example, the health check is named `Get example.com`.

```
>>>> Enter a name for the search LDAP Health Check that you want to create:
 Get example.com
```

9. Enable the new health check.

```
>>>> Configuring the 'enabled' property

Indicates whether this LDAP health check is enabled for use in the server.

Select a value for the 'enabled' property:

   1) true
   2) false

   ?) help
   c) cancel
   q) quit

Enter choice [c]: 1
```

10. Next, configure the properties of the health check. You may need to modify the `base-dn` property, as well as one or more response time thresholds for non-local external servers, accommodating WAN latency. Below is a Search LDAP Health Check for the single entry `dc=example,dc=com`, which allows non-local responses of up to 2 seconds to still be considered healthy.

```
>>>> Configure the properties of the Search LDAP Health Check

           Property                                   Value(s)
           -----------------------------------------------------------
      1)   description                                -
      2)   enabled                                    true
      3)   use-for-all-servers                        false
      4)   base-dn                                    "dc=example,dc=com"
      5)   scope                                      base-object
      6)   filter                                     (objectClass=*)
      7)   maximum-local-available-response-time      1 s
      8)   maximum-nonlocal-available-response-time   2 s
      9)   minimum-local-degraded-response-time       500 ms
      10)  minimum-nonlocal-degraded-response-time    1 s
      11)  maximum-local-degraded-response-time       10 s
      12)  maximum-nonlocal-degraded-response-time    10 s
      13)  minimum-local-unavailable-response-time    5 s
      14)  minimum-nonlocal-unavailable-response-time 5 s
      15)  allow-no-entries-returned                  true
      16)  allow-multiple-entries-returned            true
      17)  available-filter                           -
      18)  degraded-filter                            -
      19)  unavailable-filter                         -

      ?) help
      f) finish - create the new Search LDAP Health Check
      d) display the equivalent dsconfig arguments to create this object
      b) back
      q) quit
```

Configuring LDAP External Servers

The LDAP external server configuration element defines the connection, location, and health check information necessary for the Directory Proxy Server to communicate with the server properly.

PingDirectoryProxy Server includes a tool, `prepare-external-server`, for configuring communication between the Directory Proxy Server and the LDAP backend server. After you add a new LDAP external server to an existing installation, we strongly recommend that you run this tool to automatically create the user account necessary for communications. The `prepare-external-server` tool does not make configuration changes to the local Directory Proxy Server, only the external server is modified. When you run this tool, you must supply the user account and password that you specified for the Directory Proxy Server during configuration, `cn=Proxy User` by default.

Important: You should not use `cn=Directory Manager` as the account to use for communication between the Directory Proxy Server and the Directory Server. For security reasons, the account used to communicate between the Directory Proxy Server and the Directory Server should not be directly accessible by clients accessing the Directory Proxy Server. The account that you choose should meet the following criteria:

- For all server types, it should not exist in the Directory Proxy Server but only in the backend directory server instances.
- For Ping Identity Directory Server, this user should be a root user.
- For Ping Identity Directory Server, this user should not automatically inherit the default set of root privileges, but instead should have exactly the following set of privileges: `bypass-read-acl`, `config-read`, `lockdown-mode`, `proxied-auth`, and stream-values.
- For Sun Directory Servers, the account should be created below the `cn=Root DNs,cn=config` entry and the `nsSizeLimit`, `nsTimeLimit`, `nsLookThroughLimit`, and `nsIdleTimeout` values for the account should be set to -1. You also need to create access control rules to grant the user account appropriate permissions within the server. The `prepare-external-server` tool handles all of this work automatically.

About the prepare-external-server Tool

Use the `prepare-external-server` tool if you have added LDAP external servers using `dsconfig`. The `create-initial-proxy-config` tool automatically runs the `prepare-external-server` tool to configure server communications so that you do not need to invoke it separately. The `create-initial-proxy-config` tool verifies that the proxy user account exists and has the correct password and required privileges. If it detects any problems, it prompts for manager credentials to rectify them.

If you want the `prepare-external-server` tool to add the LDAP external server's certificates to the Directory Proxy Server's trust store, you must include the --proxyTrustStorePath option, and either the --proxyTrustStorePassword or the --proxyTrustStorePasswordFile option. The default location of the Directory Proxy Server trust store is `config/truststore`. The pin is encoded in the `config/truststore.pin` file.

For example, run the tool as follows to prepare an PingDirectory Server on the remote host, ds-east-01.example.com, listening on port 1389 for access by the Directory Proxy Server using the default user account `cn=Proxy User`:

```
prepare-external-server --hostname ds-east-01.example.com \
--port 1389 --baseDN dc=example,dc=com --proxyBindPassword secret
```

When the `prepare-external-server` command above is executed, it creates the `cn=Proxy User` Root DN entry as well as an access control rule in the Directory Server to grant the proxy user the proxy access right.

Note: For non-Ping Identity servers, the --baseDN argument is required for the `prepare-external-server` tool. The base DN is used to create the global ACI entries for these servers.

To Configure Server Communication Using the prepare-external-server Tool

The following example illustrates how to run the `prepare-external-server` tool to prepare a Directory Server on the remote host, ds-east-01.example.com, listening on port 1636. The Directory Server is being accessed by a Directory Proxy Server that uses the default user account `cn=Proxy User,cn=Root DNs,cn=config`. Since a password to the truststore is not provided, the truststore defined in the `--proxyTrustStorePath` is referenced in a read-only manner.

- Use the `prepare-external-server` tool to prepare the Directory Server. Follow the prompts to set up the external server.

```
$ ./PingDirectoryProxy/bin/prepare-external-server \
  --baseDN dc=example,dc=com
  --proxyBindPassword password \
  --hostname ds-east-01.example.com \
  --useSSL \
  --port 1636
  --proxyTrustStorePath /full/path/to/trust/store \
  --proxyTrustStorePassword secret
```

```
Testing connection to ds-east-01.example.com:1636 .....

Do you wish to trust the following certificate?

Certificate Subject: CN=ds-east-01.example.com, O=Example Self-Signed
 Certificate
Issuer Subject:      CN=ds-east-01.example.com, O=Example Self-Signed
 Certificate
Validity:            Thu May 21 08:02:30 CDT 2009 to Wed May 16 08:02:30 CDT
 2029

Enter 'y' to trust the certificate or 'n' to reject it.

y

The certificate was added to the local trust store

Done
Testing 'cn=Proxy User' access to ds-east-01.example.com:1636 ..... Failed
 to bind as
'cn=Proxy User'

Would you like to create or modify root user 'cn=Proxy User' so that it is
 available
for this Directory Proxy Server? (yes / no) [yes]:

Enter the DN of an account on ds-east-01.example.com:1636 with which to
 create or
manage the 'cn=Proxy User' account [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':

Created 'cn=Proxy User,cn=Root DNs,cn=config'

Testing 'cn=Proxy User' privileges ..... Done
```

To Configure an External Server Using dsconfig

1. Use the `dsconfig` tool to create and configure external servers. Then, specify the hostname, connection method, port number, and bind DN as described in previous procedures.

```
$ bin/dsconfig
```

2. In the Directory Proxy Server main menu, enter the number corresponding to external servers. Then, enter the number to create a new external server.
3. Select the type of server you want to create. This example creates a new Ping Identity Directory Server.

```
>>>> Select the type of external server that you want to create:

  1) Ping Identity DS external server
  2) JDBC external server
  3) LDAP external server
  4) Sun DS external server

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

4. Specify a name for the new external server. In this example, the external server is named east1.

```
>>>> Enter a name for the Ping Identity DS external server that you want
to create: east1
```

5. Configure the host name or IP address of the target LDAP external server.

```
Enter a value for the 'server-host-name' property:east1.example.com
```

6. Next, configure the location property of the new external server.

```
Do you want to modify the 'location' property?

   1) Leave undefined
   2) Change it to the location: East
   3) Change it to the location: West
   4) Create a new location

   ?) help
   q) quit

Enter choice [1]: 2
```

7. Next, define the bind DN and bind password.

```
Do you want to modify the 'bind-dn' property?

   1) Leave undefined
   2) Change the value

   ?) help
   q) quit

Enter choice [1]: 2

Enter a value for the 'bind-dn' property [continue]: cn=Proxy User,cn=Root
 DNs,cn=config

Enter choice [b]: 6

...

Do you want to modify the 'password' property?

   1) Leave undefined
   2) Change the value
   ?) help
   q) quit

Enter choice [1]: 2

Enter a value for the 'password' property [continue]:
Confirm the value for the 'password' property:
```

8. Enter `f` to finish the operation.

```
Enter choice [b]: f
```

```
The Ping Identity DS external server was created successfully.
```

Once you have completed adding the server, run the `prepare-external-server` tool to configure communications between the Directory Proxy Server and the Ping Identity Directory Server(s).

To Configure Authentication with a SASL External Certificate

By default, the Directory Proxy Server authenticates to the Directory Server using LDAP simple authentication (with a bind DN and a password). However, the Directory Proxy Server can be configured to use SASL EXTERNAL to authenticate to the Directory Server with a client certificate.

Have Directory Proxy Server instances installed and configured to communicate with the backend Directory Server instances using either SSL or StartTLS. After the servers are configured, perform the following steps to configure SASL EXTERNAL authentication.

1. Create a JKS keystore that includes a public and private key pair for a certificate that the Directory Proxy Server instance(s) will use to authenticate to the Directory Server instance(s). Run the following command in the instance root of one of the Directory Proxy Server instances. When prompted for a keystore password, enter a strong password to protect the certificate. When prompted for the key password, press **ENTER** to use the keystore password to protect the private key:

```
$ keytool -genkeypair \
  -keystore config/proxy-user-keystore \
  -storetype JKS \
  -keyalg RSA \
  -keysize 2048 \
  -alias proxy-user-cert \
  -dname "cn=Proxy User,cn=Root DNs,cn=config" \
  -validity 7300
```

2. Create a `config/proxy-user-keystore.pin` file that contains a single line that is the keystore password provided in the previous step.
3. If there are other Directory Proxy Server instances in the topology, copy the `proxy-user-keystore` and `proxy-user-keystore.pin` files into the `config` directory for all instances.
4. Use the following command to export the public component of the proxy user certificate to a text file:

```
$ keytool -export \
  -keystore config/proxy-user-keystore \
  -alias proxy-user-cert \
  -file config/proxy-user-cert.txt
```

5. Copy the `proxy-user-cert.txt` file into the `config` directory of all Directory Server instances. Import that certificate into each server's primary trust store by running the following command from the server root. When prompted for the keystore password, enter the password contained in the `config/truststore.pin` file. When prompted to trust the certificate, enter **yes**.

```
$ keytool -import \
  -keystore config/truststore \
  -alias proxy-user-cert \
  -file config/proxy-user-cert.txt
```

6. Update the configuration for each Directory Proxy Server instance to create a new key manager provider that will obtain its certificate from the `config/proxy-user-keystore` file. Run the following `dsconfig` command:

   ```
   $ dsconfig create-key-manager-provider \
     --provider-name "Proxy User Certificate" \
     --type file-based \
     --set enabled:true \
     --set key-store-file:config/proxy-user-keystore \
     --set key-store-type:JKS \
     --set key-store-pin-file:config/proxy-user-keystore.pin
   ```

7. Update the configuration for each LDAP external server in each Directory Proxy Server instance to use the newly-created key manager provider, and also to use SASL EXTERNAL authentication instead of LDAP simple authentication. Run the following `dsconfig` command:

   ```
   $ dsconfig set-external-server-prop \
     --server-name ds1.example.com:636 \
     --set authentication-method:external \
     --set "key-manager-provider:Proxy User Certificate"
   ```

After these changes, the Directory Proxy Server should re-establish connections to the LDAP external server and authenticate with SASL EXTERNAL. Verify that the Directory Proxy Server is still able to communicate with all backend servers by running the `bin/status` command. All of the servers listed in the "--- LDAP External Servers ---" section should be available. Review the Directory Server access log can to make sure that the BIND RESULT log messages used to authenticate the connections from the Directory Proxy Server include `authType="SASL"`, `saslMechanism="EXTERNAL"`, `resultCode=0`, and `authDN="cn=Proxy User,cn=Root DNs,cn=config"`.

Configuring Load Balancing

You can distribute the load on your Directory Proxy Server using one of the load-balancing algorithms provided with PingDirectoryProxy Server. By default, the Directory Proxy Server prefers local servers over non-local servers, unless you set the `use-location` property of the load-balancing algorithm to false. Within a given location, the Directory Proxy Server prefers available servers over degraded servers. This means that if at all possible, the Directory Proxy Server sends requests to servers that are local and available before considering selecting any server that is non-local or degraded.

Note: If the `use-location` property is set to true, then the load is balanced only among available external servers in the same location. If no external servers are available in the same location, the Directory Proxy Server will attempt to use available servers in the first preferred failover location, and so on. The failover based on no external servers with AVAILABLE health state can be customized to allow the Directory Proxy Server to prefer local DEGRADED health servers to servers in a failover location. See the PingDirectoryProxy Server Reference Guide for more information on the `prefer-degraded-servers-over-failover` property.

The Directory Proxy Server provides the following load-balancing algorithms:

- **Failover load balancing**. This algorithm forwards requests to servers in a given order, optionally taking the location into account. If the preferred server is not available, then it will fail over to the alternate server in a predefined order. This balancing method can be useful if certain operations, such as LDAP writes, need to be forwarded to a primary external server, with secondary external servers defined for failover if necessary.
- **Fewest operations load balancing**. This algorithm forwards requests to the backend server with the fewest operations currently in progress and tends to exhibit the best performance.
- **Health weighted load balancing**. This algorithm assigns weights to servers based on their health scores and, optionally, their locations. For example, servers with a higher health check score will receive a higher proportion of the requests than servers with lower health check scores.
- **Single server load balancing**. This algorithm forwards all operations to a single external server that you specify.

- **Weighted load balancing**. This algorithm uses statically defined weights for sets of servers to divide load among external servers. External servers are grouped into weighted sets, the values of which, when added to all of the weighted sets for the load balancing algorithm, represent a percentage of the load the external servers should receive.
- **Criteria based load balancing**. This algorithm allows you to balance your load across a server topology depending on the types of operations received or the client issuing the request.

  For example, ds1 and ds2 are assigned to a weighted set named Set-80 and assigned the weight 80. The external servers ds3 and ds4 are assigned to the weighted set Set-20 and assigned the weight 20. When both sets, Set-80 and Set-20, are assigned to the load balancing algorithm, 80 percent of the load will be forwarded to ds1 and ds2, while the remaining 20 percent will be forwarded to ds3 and ds4.

To Configure Load Balancing Using dsconfig

1. Use the `dsconfig` tool to create and configure a load-balancing algorithm.

   ```
   $ bin/dsconfig
   ```

   Specify the hostname, connection method, port number, and bind DN as described in previous procedures.
2. In the Directory Proxy Server main menu, enter the number associated with load-balancing algorithms.
3. Select an existing load-balancing algorithm to use as a template or select `n` to create a new load-balancing algorithm from scratch.

   ```
   >>>>Choose how to create the new Load Balancing Algorithm:

   n) new Load Balancing Algorithm created from scratch
   t) use an existing Load Balancing Algorithm as a template
   b) back
   q) quit

   Enter a choice [n]: n
   ```
4. Select the type of load-balancing algorithm that you want to create. Depending on type of algorithm you select, you will be guided through a series of configuration properties, such as providing a name and selecting an LDAP external server.

   ```
   >>>> Select the type of Load Balancing Algorithm that you want to
   create:

       1)  Failover Load Balancing Algorithm
       2)  Fewest Operations Load Balancing Algorithm
       3)  Health Weighted Load Balancing Algorithm
       4)  Single Server Load Balancing Algorithm
       5)  Weighted Load Balancing Algorithm

       ?)  help
       c)  cancel
       q)  quit

   Enter choice [c]: 3
   ```
5. Review the configuration properties for your new load-balancing algorithm. If you are satisfied, enter `f` to finish.

Configuring Criteria-Based Load-Balancing Algorithms

You can configure alternate load-balancing algorithms that determine how they function according to request or connection criteria. These algorithms allow you to balance your load across a server topology depending on the types of operations received or the client issuing the request. They are called criteria-based load-balancing algorithms and are configured using at least one connection criteria or request criteria. For example, you can configure criteria-based load-balancing algorithms to accomplish the following:

- Route write operations to a single server from a set of replicated servers, to prevent replication conflicts, while load balancing all other operations across the full set of servers.
- Route all operations from a specific client to a single server in a set of replicated servers, eliminating errors that arise from replication latency, while load balancing operations from other clients across the full set of servers. This configuration is useful for certain provisioning applications that need to write and then immediately read the same data.

When a request is received, the proxying request processor first iterates through all of the criteria-based load-balancing algorithms in the order in which they are listed, to determine whether the request matches the associated criteria. If there is a match, then the criteria-based load-balancing algorithm is selected. If there is not a match, then the default load-balancing algorithm is used.

Preferring Failover LBA for Write Operations

An administrator can configure the Directory Proxy Server to use Criteria-Based Load-Balancing Algorithms to strike a balance between providing a consistent view of directory server data for applications that require it and taking advantage of all servers in a topology for handling read-only operations, such as search and bind. The flexible configuration model supports a wide range of criteria for choosing which Load-Balancing Algorithm to use for each operation. In most Directory Proxy Server deployments, Ping Identity recommends using a Failover Load-Balancing Algorithm for at least ADD, DELETE, and MODIFY-DN operations if not for all types of write operations.

Each Proxying Request Processor configured in the Directory Proxy Server uses a Load-Balancing Algorithm to choose which Directory Server to use for a particular operation. The Load-Balancing Algorithm takes several factors into account when choosing a server:

The availability of the directory servers.

The location of the directory servers. By default Load-Balancing Algorithms prefer directory servers in the same location as the Directory Proxy Server.

Whether the Directory Server is degraded for any reason, such as having a Local DB Index being rebuilt.

The result of configured Health Checks. For instance, a server with a small replication backlog can be preferred over one with a larger backlog.

Recent operation routing history.

How these factors are used depends on the specific Load-Balancing Algorithm. The two most commonly used Load-Balancing Algorithms are the Failover Load-Balancing Algorithm and the Fewest Operations Load-Balancing Algorithm. These two algorithms are similar when determining which Directory Servers are the possible candidates for a specific operation. The algorithms use the same criteria to determine server availability and health, and by default they will prefer Directory Servers in the same location as the Directory Proxy Server. However, they differ in the criteria they use to choose between available servers.

The Failover Load-Balancing Algorithm will send all operations to a single server until it is unavailable, and then it will send all operations to the next preferred server, and so on. This algorithm provides the most consistent view of the topology to clients because all clients (at least those in the same location as the Directory Proxy Server) will see the same, up-to-date view of the data, but it leaves unused capacity in the failover instances since most topologies include multiple Directory Server replicas within each data center.

On the other hand, the Fewest Operations Load-Balancing Algorithm does the best job of efficiently distributing traffic among multiple servers since it chooses to send each operation to the server that has the fewest number of outstanding operations--that is, the server from the Directory Proxy Server's point of view that is the least busy. (Note: the Fewest Operations Load-Balancing Algorithm routes traffic to the least loaded server, which in a lightly-loaded environment can result in an imbalance since the first server in the list of configured servers is more likely to receive a request.) This algorithm naturally routes to servers that are more responsive as well as limiting the impact of servers that have become unreachable. However, this implies that consecutive operations that depend on each other can be routed to different Directory Servers, which can cause issues for some types of clients:

If two entries are added in quick succession where the first entry is the parent of the second in the LDAP hierarchy, then the addition of the child entry could fail if that operation is routed to a different Directory Server instance than the first ADD operation, and this happens within the replication latency.

Some clients add or modify an entry and then immediately read the entry back from the server, expecting to see the updates reflected in the entry.

In these situations, it is desirable to configure the <keyword keyref="PROXY_SERVER_BASE_NAME"/> to route dependent requests to the same server.

The server affinity feature (see *Configuring Server Affinity*) achieves this in some environments but not in all because the affinity is tracked independently by each Directory Proxy Server instance, and some clients send requests to multiple proxies. It is common for a client to not connect to the Directory Proxy Servers directly but instead to connect through a network load balancer, which in turn opens connections to the Directory Proxy Servers. Each individual client connection will be established to a single Directory Proxy Server so that operations on that connection will be routed to the same Directory Proxy Server, and server affinity configured within the Directory Proxy Server will ensure those operations will be routed to the same Directory Server. However, many clients establish a pool of connections that are reused across operations, and within this pool, connections will be established through the load balancer to different Directory Proxy Servers. Dependent operations sent on different connections could then be routed to different Directory Proxy Servers, and then on to different Directory Servers.

A Failover Load-Balancing Algorithm addresses this issue by routing all requests to a single server, but that leaves unused search capacity on the other instances. A Criteria Based Load-Balancing Algorithm enables the proxy to route certain types of requests (or requests from certain clients) using a different Load-Balancing Algorithm than the default. For instance, all write operations (i.e., ADD, DELETE, MODIFY, and MODIFY-DN) could be routed using a Failover Load-Balancing Algorithm, while all other operations (bind, search, and compare) use a Fewest Operations Load-Balancing Algorithm. And in addition, if there are clients that are particularly sensitive to reading entries immediately after modifying them, additional Connection Criteria can be specified to all operations from those clients using the Failover Load-Balancing Algorithm. Note that, routing all write requests to a single server in a location instead of evenly across servers does not limit the overall throughput of the system since all servers ultimately have to process all write operations either from the client directly or via replication.

Another benefit of using the Failover Load-Balancing Algorithm for write operations is reducing replication conflicts. The Ping Identity Directory Server follows the traditional LDAP replication model of eventual consistency. This provides very high availability for handling write traffic even in the presence of network partitions, but it can lead to replication conflicts. Replication conflicts involving modify operations can be automatically resolved, leaving the servers in a consistent state where each attribute on each entry reflects the most recent update to that attribute. However, conflicts involving ADD, DELETE, and MODIFY-DN operations cannot always be resolved automatically and can require manual involvement from an administrator. By routing all write operations (or at least ADD, DELETE, and MODIFY-DN operations) to a single server, replication conflicts can be avoided.

There are a few points to consider when using a Failover Load-Balancing Algorithm:

- When using the Failover Load-Balancing Algorithm in a configuration with multiple locations, the Load-Balancing Algorithm will fail over between local instances before failing over to servers in a remote location. The list of servers in the `backend-server` configuration property of the Load-Balancing Algorithm should be ordered such that preferred local servers should appear before failover local servers, but the relative order of servers in different locations is unimportant as the `preferred-failover-location` of the Directory Proxy Server's configuration is used to decide which remote location to fail over to. It is also advisable that the order of local servers match the `gateway-priority` configuration settings of the "Replication Server" configuration object on the Directory Server instances. This can reduce the WAN replication delay because the Directory Proxy Server will then prefer to send writes to the Directory Server with the WAN Gateway role, avoiding an extra hop to the remote locations.
- For Directory Proxy Server configurations that include multiple Proxying Request Processors, including Entry-Balancing environments, each Proxying Request Processor should be updated to include its own Criteria-Based Load-Balancing Algorithm.

To Route Operations to a Single Server

The following example shows how to extend a Directory Proxy Server's configuration to use a Criteria Based Load Balancing Algorithm to route all write requests to a single server using a Failover Load Balancing Algorithm. The approach outlined here can easily be extended to support alternate criteria as well as more complex topologies using multiple locations or Entry Balancing.

This example uses a simple deployment of a Directory Proxy Server fronting three Directory Servers: ds1.example.com, ds2.example.com, and ds3.example.com.

Once these configurations changes are applied, the Directory Proxy Server will route all write operations to ds1.example.com as long as it is available and then to ds2.example.com if it is not, while routing other types of operations, such as searches and binds, to all three servers using the Fewest Operations Load Balancing Algorithm.

1. First, create a location.

   ```
   dsconfig create-location --location-name Austin
   ```
2. Update the failover location for your server.

   ```
   dsconfig set-location-prop --location-name Austin
   ```
3. Set the location as a global configuration property.

   ```
   dsconfig set-global-configuration-prop --set location:Austin
   ```
4. Set up the health checks for each external server.

   ```
   dsconfig create-ldap-health-check \
   --check-name ds1.example.com:389_dc_example_dc_com-search-health-check \
   --type search --set enabled:true --set base-dn:dc=example,dc=com

   dsconfig create-ldap-health-check \
   --check-name ds2.example.com:389_dc_example_dc_com-search-health-check \
   --type search --set enabled:true --set base-dn:dc=example,dc=com

   dsconfig create-ldap-health-check \
   --check-name ds3.example.com:389_dc_example_dc_com-search-health-check \
   --type search --set enabled:true --set base-dn:dc=example,dc=com
   ```
5. Create the external servers.

   ```
   dsconfig create-external-server --server-name ds1.example.com:389 \
   --type ping identity-ds \
   --set server-host-name:ds1.example.com --set server-port:389 \
   --set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
   --set password:AADoPkhx22qpiBQJ7T0X4wH7 \
   --set health-check:ds1.example.com:389_dc_example_dc_com-search-health-check


   dsconfig create-external-server --server-name ds2.example.com:389 \
   --type ping identity-ds \
   --set server-host-name:ds2.example.com --set server-port:389 \
   --set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
   --set password:AAAoVqVYsEavey80T0QfR60I \
   --set health-check:ds2.example.com:389_dc_example_dc_com-search-health-check


   dsconfig create-external-server --server-name ds3.example.com:389 \
   --type ping identity-ds \
   --set server-host-name:ds3.example.com --set server-port:389 \
   --set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
   --set password:AADOkveb0TtYR9xpkVrNgMtF \
   --set health-check:ds3.example.com:389_dc_example_dc_com-search-health-check
   ```
6. Create a Load Balancing Algorithm for `dc=example,dc=com`.

   ```
   dsconfig create-load-balancing-algorithm \
   --algorithm-name dc_example_dc_com-fewest-operations \
   --type fewest-operations --set  enabled:true \
   --set backend-server:ds1.example.com:389 \
   --set backend-server:ds2.example.com:389 \
   ```

```
--set backend-server:ds3.example.com:389
```

7. Create a Request Processor for `dc=example,dc=com`.

```
dsconfig create-request-processor \
--processor-name dc_example_dc_com-req-processor \
--type proxying \
--set load-balancing-algorithm:dc_example_dc_com-fewest-operations
```

8. Create a Subtree View for `dc=example,dc=com`.

```
dsconfig create-subtree-view \
--view-name dc_example_dc_com-view \
--set base-dn:dc=example,dc=com \
--set request-processor:dc_example_dc_com-req-processor
```

9. Update the client connection policy for dc=example,dc=com.

```
dsconfig set-client-connection-policy-prop \
--policy-name default \
--add subtree-view:dc_example_dc_com-view
```

10. Create a new Request Criteria object to match all write operations.

```
dsconfig create-request-criteria \
--criteria-name any-write \
--type simple --set "description:All Write Operations" \
--set operation-type:add --set operation-type:delete \
--set operation-type:modify --set operation-type:modify-dn
```

11. Create a new Failover Load Balancing Algorithm listing the servers that should be included. Note the order that the servers are listed here is the failover order between servers.

```
dsconfig create-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-failover \
--type failover --set enabled:true \
--set backend-server:ds1.example.com:389 \
--set backend-server:ds2.example.com:389 \
--set backend-server:ds3.example.com:389
```

12. Tie the Request Criteria and the Failover Load Balancing Algorithm together into a Criteria Based Load Balancing Algorithm.

```
dsconfig create-criteria-based-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-write-traffic-lba \
--set "description:Failover LBA For All Write Traffic" \
--set request-criteria:any-write \
--set load-balancing-algorithm:dc_example_dc_com-failover
```

13. Update the Proxying Request Processor to use the Criteria Based Load Balancing Algorithm.

```
dsconfig set-request-processor-prop \
--processor-name dc_example_dc_com-req-processor \
--set criteria-based-load-balancing-algorithm:dc_example_dc_com-write-
traffic-lba
```

To Route Operations from a Single Client to a Specific Set of Servers

To create a type of server affinity, where all operations from a single client are routed to a specific set of servers, follow a similar process as in the previous use case. Instead of request criteria, configure connection criteria. These connection criteria identify clients that could be adversely affected by replication latency. These clients will use the Failover Load Balancing Algorithm rather than the default Fewest Operations Load Balancing Algorithm.

For example, an administrative tool includes a "delete user" function. If the application immediately re-queries the directory for an updated list of users, the just-deleted entry must not be included. To configure a criteria-based load balancing algorithm to support this use case, perform the following:

- Create a failover load balancing algorithm that lists the same set of servers as the existing fewest operation load balancing algorithm.
- Create connection criteria that match the clients for which failover load balancing should be applied, rather than fewest operations load balancing.
- Create a criteria-based load balancing algorithm that references the two configuration objects created in the previous steps.
- Assign the new load balancing algorithm to the proxying request processor.

The following procedure provides examples of each of these steps.

1. Create the new failover load balancing algorithm using `dsconfig` as follows:

```
dsconfig create-load-balancing-algorithm \
  --algorithm-name client_one_routing_algorithm \
  --type failover --set enabled:true \
  --set backend-server:east1.example.com:389 \
  --set backend-server:east2.example.com:389
```

2. To route operations from a single client to a single server in a set of replicated servers, create connection criteria using `dsconfig` as follows:

```
dsconfig create-connection-criteria \
  --criteria-name "Client One"  --type simple \
  --set included-user-base-dn:cn=Client One,ou=Apps,dc=example,dc=com
```

3. Configure a criteria-based load balancing algorithm and assign it to the proxying request processor. Use the load balancing algorithm and connection criteria created in the previous steps:

```
dsconfig create-criteria-based-load-balancing-algorithm \
  --algorithm-name dc_example_dc_com-client-operations \
  --set load-balancing-algorithm:dc_example_dc_com-failover \
  --set "request-criteria:Client One Requests"
```

4. Assign the new criteria-based load balancing algorithm to the proxying request processor using `dsconfig` as follows:

```
dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add criteria-based-load-balancing-agorithm:dc_example_dc_com-client-
operations
```

Understanding Failover and Recovery

Once a previously degraded or unavailable server has recovered, it should be eligible to start receiving traffic within the time configured for the health-check-frequency property, 30 seconds by default. However, failover and recovery also depend on the load-balancing algorithm in use.

The load-balancing algorithm provides an ordered list of servers to check, with the number of servers in the list based on the maximum number of retry attempts. The server checks to see if affinity should be used and, if so, whether an affinity is set for that load-balancing algorithm. If there is an affinity to a particular server and that server is classified as available, then that server will always be the first in the list.

Next, the Directory Proxy Server creates a two-dimensional matrix of servers based on the health check state (with available preferred over degraded and unavailable not considered at all) and location (with backend servers in the same location as the Directory Proxy Server most preferred, then servers in the first failover location, then the second, and so on). Within each of these sets, and ideally at least one server in the local data center is classified as available, the load-balancing algorithm selects the servers in the order of most preferred to least preferred based on whatever logic the load-balancing algorithm uses. The load-balancing algorithm keeps selecting servers until enough of them have been selected to satisfy the maximum number of possible retries.

The load-balancing algorithm includes a configuration option that allows you to decide whether to prefer location over availability and vice-versa. For example, is a local degraded server more or less preferred than a remote available server? By default, the algorithm will prefer available servers over degraded ones, even if it has to go to another data center to access them. You can change the load-balancing algorithm to try to stay in the same data center if at least one server is not unavailable.

The Directory Proxy Server does both proactive and reactive health checking. With proactive health checking, the Directory Proxy Server will periodically (by default, every 30 seconds), run a full set of tests against each backend server. The result of these tests will be usd to determine the overall health check state (available, degraded, or unavailable) and score (and integer value from 10 to 0). With reactive health checking, the Directory Proxy Server may kick off a lesser set of health checks against a server if an operation forwarded to that server did not complete successfully.

Proactive health checking can be used to promote and demote the health of a server, but reactive health checking can only be used to demote the health of a server. As a result, if a server is determined to be unavailable, then it will remain that way until a subsequence proactive health check determines that it has recovered. If a server is determined to be degraded, it may not become available until the next proactive health check, but it could be downgraded to unavailable by a reactive check if other failures are encountered against that server.

Both proactive and reactive health check assignments take effect immediately and will be considered for all subsequent requests routed to the load-balancing algorithm. If a a server is considered degraded, then it will immediately be considered less desirable than available servers in the same data center, and possibly less desirable than available servers in more remote data centers. If a server is considered unavailable, then it will not be eligible to be selected until it is reclassified as available or degraded.

Configuring Proxy Transformations

The PingDirectoryProxy Server provides proxy transformations to alter the contents of client requests as they are sent from the client to the LDAP external server. Proxy transformations can also be used to alter the responses sent back from the server to the client, including altering or omitting search result entries. The Directory Proxy Server provides the following types of data transformations:

- **Attribute mapping**. This transformation rewrites client requests so that references to one attribute type may be replaced with an alternate attribute type. The Directory Proxy Server can perform extensive replacements, including attribute names used in DNs and attribute names encoded in the values of a number of different controls and extended operations. For example, a client requests a `userid` attribute, which is replaced with `uid` before being forwarded on to the backend server. This mapping applies in reverse for the response returned to the client.
- **Default value**. This transformation instructs the Directory Proxy Server to include a static attribute value in search results being sent back to the client, in ADD requests being forwarded to an external server, or both. For example, a value of "marketing" for `businessCategory` could be returned for all search results under the base DN `ou=marketing,dc=example,dc=com`.
- **DN mapping**. This transformation rewrites client requests so that references to entries below a specified DN will be mapped to appear below another DN. For example, references to entries below `o=example.com` could be rewritten so that they are below `dc=example,dc=com` instead. The mapping applies in reverse for the response returned to the client.
- **Groovy scripted**. This custom transformation is written in Groovy and does not need to be compiled, though they use the Server SDK. These scripts make it possible to alter requests and responses in ways not available using the transformations provided with the Directory Proxy Server.
- **Suppress attribute**. This proxy transformation allows you to exclude a specified attribute from search result entries. It also provides the ability to reject add, compare, modify, modify DN, or search requests if they attempt to reference the target attribute.
- **Suppress entry**. This proxy transformation allows you to exclude any entries that match a specified filter from a set of search results. Search requests are transformed so that the original filter will be ANDed with a NOT filter containing the exclude filter. For example, if the suppression filter is `"(objectClass=secretEntry)"`, then a search request with a filter of `"(uid=john.doe)"` will be transformed so that it has a filter of `"(&(uid=john.doe)(!(objectClass=secretEntry)))"`.
- **Simple to external bind**. This proxy transformation may be used to intercept a simple bind request and instead process the bind as a SASL EXTERNAL bind. If the SASL EXTERNAL bind fails, then the original simple bind request may or may not be processed, depending on how you configure the server.

- **Third-party scripted**. This custom transformation is created using the Server SDK, making it possible to alter requests and responses in ways not available using the transformations provided with the Directory Proxy Server.

To Configure Proxy Transformations Using dsconfig

1. Use the `dsconfig` tool to create and configure a proxy transformation.

   ```
   $ bin/dsconfig
   ```
2. Enter the connection parameters (for example, hostname, port, bind DN and bind DN password).
3. In the Directory Proxy Server main menu, enter the number associated with proxy transformations. On the Proxy Transformation menu, enter the number to create a new proxy transformation.
4. Select the type of proxy transformation you want to create. In this example, we create an attribute mapping transformation. Then, enter a name for the new transformation.

   ```
    >>>> Enter a name for the Attribute Mapping Proxy Transformation that you
   want to create: userid-to-uid
   ```
5. Indicate whether you want the transformation to be enabled by default.

   ```
   Select a value for the 'enabled' property:

     1)  true
     2)  false

     ?)  help
     c)  cancel
     q)  quit

   Enter choice [c]: 1
   ```
6. Specify the name of the client attribute that you want to remap to a target attribute. Note that this attribute must not be equal to the target attribute.

   ```
   Enter a value for the 'source-attribute' property: userid
   ```
7. Specify the name of the target attribute to which the client attribute should be mapped.

   ```
   Enter a value for the 'target-attribute' property: uid
   ```
8. The properties of your new proxy transformation are displayed. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the proxy transformation.

   ```
   Enter choice [b]: f
   ```

   The transformation now needs to be assigned to a request processor. To create an initial request processor, see the next section.

Configuring Request Processors

A request processor is responsible for handling client requests by passing the request through a load-balancing algorithm or one or more subordinate request processors. The request processor is also the Directory Proxy Server component that performs proxy transformations. You can create one of the following types of request processors:

- **Proxying request processor**. This request processor is responsible for passing allowed operations through a load balancing algorithm. Proxy transformations can be applied to requests and responses that are processed. Multiple servers may be configured to provide high availability and load balancing, and various transformations may be applied to the requests and responses that are processed.
- **Entry-balancing request processor**. This request processor is used to distribute entries under a common parent entry among multiple backend sets. A backend set is a collection of replicated directory servers that contain identical portions of the data. This request processor uses multiple, subordinate proxying request processors

to process operations and maintains in-memory indexes to speed the processing of specific search and modify operations.

- **Failover request processor**. This request processor performs ordered failover between subordinate proxying processors, sometimes with different behavior for different types of operations.

To Configure Request Processors Using dsconfig

1. Use the `dsconfig` tool to create and configure a request processor.

   ```
   $ bin/dsconfig
   ```

2. Specify the hostname, connection method, port number, and bind DN as described in previous procedures.
3. In the Directory Proxy Server main menu, enter the number associated with Request Processor configuration and select the option to create a new Request Processor.
4. Select an existing request processor to use as a template for creating a new one or enter `n` to create one from scratch. In this example, we create a new proxying request processor from scratch. You will be required to choose an existing load balancing algorithm or create a new one to complete the create of the request processor. Below is the configuration of the proxying request processor after selection of the load balancing algorithm.

   ```
   Property                     Value(s)
   --------------------------------------------------------------
   1)   description             -
   2)   enabled                 true
   3)   allowed-operation       abandon, add, bind, compare, delete,
                                extended, modify, modify-dn, search
   4)   load-balancing-algorithm dc_example_dc_com-fewest-operations
   5)   transformation          -
   6)   referral-behavior       pass-through
   7)   supported-control       account-usable, assertion,
                                authorization-identity,
                                get-authorization-entry,
                                get-effective-rights, get-server-id,
                                ignore-no-user-modification,
                                intermediate-client, manage-dsa-it,
                                matched-values, no-op, password-policy,
                                permissive-modify, post-read, pre-read,
                                proxied-authorization-v1,
                                proxied-authorization-v2,
                                real-attributes-only, retain-identity,
                                subentries, subtree-delete,
                                virtual-attributes-only
   8)   supported-control-oid   -

   ?)   help
   f)   finish - create the new Proxying Request Processor
   d)   display the equivalent dsconfig arguments to create this object
   b)   back
   q)   quit
   ```

5. Review the configuration properties of the new request processor. If you are satisfied, enter `f` to finish. For the request processor to be used, it must be associated with a subtree view.

To Pass LDAP Controls with the Proxying Request Processor

If your deployment does not use entry balancing and requires the use of LDAP controls not defined in the request processor's supported-control property, configure the Directory Proxy Server to forward these controls correctly. This is done by configuring the `supported-control-oid` property to define the request OID of the LDAP control. The Directory Proxy Server updates the root DSE `supportedControl` attribute with the values entered for the `supported-control-oid` property.

Configuring Server Affinity

The Directory Proxy Server supports the ability to forward a sequence of requests to the same external server if specific conditions are met. This feature, called server affinity, is applied by the load balancing algorithms. The following server affinity methods are available in the Directory Proxy Server:

- **Client Connection**. Requests from the same Directory Proxy Server client connection are consistently routed to the same external server.
- **Client IP**. Directory Proxy Server client requests coming from the same client IP address are routed to the same external server.
- **Bind DN**. Requests from all client connections authenticated as the same bind DN are routed to the same external server.

For each algorithm, you can specify the set of operations for which an affinity will be established, as well as the set of operations for which affinity will be used. Affinity assignments have a time-out value so that they are in effect for some period of time after the last operation that may cause the affinity to be set or updated.

To Configure Server Affinity

In this example, we create a bind DN server affinity provider for any client requesting write operations to have subsequent requests, whether read or write, forwarded to the same external server. The affinity period will last for 30 seconds after the last write request.

1. Use the `dsconfig` tool to configure server affinity. Specify the hostname, connection method, port number, and bind DN as described in previous procedures.

```
$ bin/dsconfig
```

2. In the Directory Proxy Server main menu, enter the number associated with server affinity provider configuration
3. On the Server Affinity menu, enter the number corresponding to creating a new server affinity provider.
4. Enter a name for your new server affinity provider.

```
>>>> Enter a name for the Bind DN Server Affinity Provider
that you want to create: Affinity for Writing Applications
```

5. Indicate whether you want the server affinity provider to be enabled for use by the Directory Proxy Server. In this example, enter `1` to enable to the server affinity provider.
6. Next, configure the properties of the server affinity provider. For example, you can customize the types of operations for which affinity may be set and the types of operations for which affinity may be used, as well as the length of time for which the affinity should persist. This example illustrates the properties of the bind DN server affinity provider.

```
>>>> >>>> Configure the properties of the Bind DN Server Affinity Provider

        Property                  Value(s)
        ----------------------------------------------------------
    1)  description               -
    2)  enable                    true
    3)  affinity-duration         30 s
    4)  set-affinity-operation    add, delete, modify, modify-dn
    5)  use-affinity-operation    add, bind, compare, delete, modify,
                                  modify-dn, search

    ?)  help
    f)  finish - create the new Bind DN Server Affinity Provider
    d)  display the equivalent dsconfig arguments to create this object
    b)  back
    q)  quit

Enter choice [b]:
```

7. Review the properties of the server affinity provider. If you are satisfied, enter `f` to finish. Once defined, the affinity provider can now be assigned to a load balancing algorithm.

Configuring Subtree Views

You provide clients access to a specific portion of the DIT creating a subtree view and assigning it to a client connection policy. You can configure subtree views from the command line or using the Administrative Console.

When you create a subtree view, you provide the following information to configure its properties:

- Subtree view name
- Base DN managed by the subtree view
- Request processor used by the subtree view to route requests. If one does not exist already, you will create a new one.

To Configure Subtree View

1. Use `dsconfig` to configure a subtree view.
2. In the Directory Proxy Server main menu, enter the number associated with subtree view configuration
3. In the Subtree View menu, enter the number corresponding to creating a new subtree view.
4. Enter a name for the subtree view.
5. Enter the base DN of the subtree managed by this subtree view.

```
Enter a value for the ‘base-dn’ property:dc=example,dc=com
```

6. Select a request processor for this subtree view to route requests or make the appropriate selection to create a new one.

```
Select a Request Processor for the ‘request-processor’ property:

1) dc_example_dc_com-req-processor
2) Create a new Request Processor

?) help
c) cancel
q) quit

Enter choice [c]: 1
```

7. Review the properties of the subtree view. If you are satisfied, enter `f` to finish.

```
>>>> Configure the properties of the Subtree View
 >>>> via creating 'example.com' Subtree View

        Property           Value(s)
        --------------------------------------------------
    1)  description        -
    2)  base-dn            "dc=example,dc=com"
    3)  request-processor  dc_example_dc_com-req-processor

    ?)  help
    f)  finish - create the new Subtree View
    d)  display the equivalent dsconfig arguments to create
        this object
    b)  back
    q)  quit
```

Once configured, you can assign one or more subtree views to any client connection policies.

Configuring Client Connection Policies

Client connection policies help distinguish what portions of the DIT the client can access. They also enforce restrictions on what clients can do in the server. A client connection policy specifies criteria for membership based on information about the client connection, including client address, protocol, communication security, and authentication state and identity. The client connection policy, however, does not control membership based on the type of request being made.

Every client connection is associated with exactly one client connection policy at any given time, which is assigned to the client when the connection is established. The choice of which client connection policy to use will be reevaluated when the client attempts a bind to change its authentication state or uses the StartTLS extended operation to convert an insecure connection to a secure one. Any changes you make to the client connection policy do not apply to existing connections. The changes only apply to new connections.

Client connections are always unauthenticated when they are first established. If you plan to configure a policy based on authentication, you must define at least one client connection policy with criteria that match unauthenticated connections.

Once a client has been assigned to a policy, you can determine what operations they can perform. For example, your policy might allow only SASL bind operations. Client connection policies are also associated with one or more subtree views, which determine the portions of the DIT a particular client can access. For example, you might configure a policy that prevents users connecting over the extranet from accessing configuration information. The client connection policy is evaluated in addition to access control, so even a root user connecting over the extranet would not have access to the configuration information.

Understanding the Client Connection Policy

Client connection policies are based on two things:

- **Connection criteria**. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used in other instances when the server needs to perform matching based on connection-level properties, such as filtered logging. A single connection can match multiple connection criteria definitions.
- **Evaluation order index**. If multiple client connection policies are defined in the server, then each of them must have a unique value for the **evaluation-order-index** property. The client connection policies are evaluated in order of ascending evaluation order index. If a client connection does not match the criteria for any defined client connection policy, then that connection will be terminated.

If the connection policy matches a connection, then the connection is assigned to that policy and no further evaluation occurs. If, after evaluating all of the defined client connection policies, no match is found, the connection is terminated.

When a Client Connection Policy is Assigned

A client connection policy can be associated with a client connection at the following times:

- When the connection is initially established. This association occurs exactly once for each client connection.
- After completing processing for a StartTLS operation. This association occurs at most once for a client connection, because StartTLS cannot be used more than once on a particular connection. You also may not stop using StartTLS while keeping the connection active.
- After completing processing for a bind operation. This association occurs zero or more times for a client connection, because the bind request can be processed many times on a given connection.

StartTLS and bind requests will be subject to whatever constraints are defined for the client connection policy that is associated with the client connection at the time that the request is received. Once they have completed, then subsequent operations will be subject to the constraints of the new client connection policy assigned to that client connection. This policy may or may not be the same client connection policy that was associated with the connection before the operation was processed. That is, any policy changes do not apply to existing connections and will be applicable when the client reconnects.

All other types of operations will be subject to whatever constraints are defined for the client connection policy used by the client connection at the time that the request is received. The client connection policy assigned to a connection never changes as a result of processing any operation other than a bind or StartTLS. So, the server will not re-evaluate the client connection policy for the connection in the course of processing an operation. For example, the client connection policy will never be re-evaluated for a search operation.

Restricting the Type of Search Filter Used by Clients

You can restrict the types of search filters that a given client may be allowed to use to prevent the use of potentially expensive filters, like range or substring searches. You can use the `allowed-filter-type` property to provide a list of filter types that may be included in the search requests from clients associated with the client connection policy. This setting will only be used if search is included in the set of allowed operation types. This restriction will only be applied to searches with a scope other than `baseObject`, such as searches with a scope of `singleLevel`, `wholeSubtree`, or `subordinateSubtree`.

The `minimum-substring-length` property can be used to specify the minimum number of non-wildcard characters in a substring filter. Any attempt to use a substring search with an element containing fewer than this number of bytes will be rejected. For example, the server can be configured to reject filters like `"(cn=a*)"` and `"(cn=ab*)"`, but to allow `"(cn=abcde*)"`. This property setting will only be used if search is included in the set of allowed operation types and at least one of `sub-initial`, `sub-any`, or `sub-final` is included in the set of allowed filter types.

There are two primary benefits to enforcing a minimum substring length:

- Allowing very short substrings can require the server to perform more expensive processing. The search requires a lot more server effort to assemble a candidate entry list for short substrings because the server has to examine a lot more index keys.
- Allowing very short substrings makes it easier for a client to put together a series of requests to retrieve all the data from the server (a process known as "trawling"). If a malicious user wants to obtain all the data from the server, then it is easier to issue 26 requests like `"(cn=a*)"`, `"(cn=b*)"`, `"(cn=c*)"`, ..., `"(cn=z*)"` than if the user is required to do something like `"(cn=aaaaa*)"`, `"(cn=aaaab*)"`, `"(cn=aaaac*)"`, ..., `"(cn=zzzzz*)"`.

Defining Request Criteria

The client connection policy provides several properties that allow you to define the kinds of requests that it can issue. The `required-operation-request-criteria` property causes the server to reject any requests that do not match the referenced request criteria. The `prohibited-operation-request-criteria` property causes the server to reject any request that matches the referenced request criteria.

Setting Resource Limits

A client connection policy can specify resource limits, helping to ensure that no single client monopolizes server resources. The resource limits are applied in addition to any global configuration resource limits. In other words, a client connection policy cannot grant additional resources beyond what is set in the global configuration. If a client connection exceeds either a globally-defined limit or a policy limit, then it is terminated.

Note: The Directory Proxy Server’s global configuration can enforce limits on the number of concurrent connections that can be established in the following ways:

- Limit the total number of concurrent connections to the server.
- Limit the total number of concurrent connections from the same IP address.
- Limit the total number of concurrent connections authenticated as the same bind DN.

Defining the Operation Rate

You can configure the maximum operation rate for individual client connections as well as collectively for all connections associated with a client connection policy. If the operation rate limit is exceeded, the Directory Proxy Server may either reject the operation or terminate the connection. You can define multiple rate limit values, making it possible to fine tune limits for both a long term average operation rate and short term operation bursts. For example, you can define a limit of one thousand operations per second and one million operations per day, which works out to an average of less than twelve operations per second, but with bursts of up to one thousand operations per second.

Rate limit strings should be specified as a maximum count followed by a slash and a duration. The count portion must contain an integer, and may be followed by a multiplier of k (to indicate that the integer should be interpreted as thousands), m (to indicate that the integer should be interpreted as millions), or g (to indicate that the integer should be interpreted as billions). The duration portion must contain a time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), and may be preceded by an integer to specify a quantity for that unit.

For example, the following are valid rate limit strings:

1/s (no more than one operation over a one-second interval)
10K/5h (no more than ten thousand operations over a five-hour interval)
5m/2d (no more than five million operations over a two-day interval)

You can provide time units in many different formats. For example, a unit of seconds can be signified using s, sec, sect, second, and seconds.

Client Connection Policy Deployment Example

In this example scenario, we assume the following:

Two external LDAP clients are allowed to bind to the Directory Proxy Server.
Client 1 should be allowed to open only 1 connection to the server.
Client 2 should be allowed to open up to 5 connections to the server.

Defining the Connection Policies

We need to set a per-client connection policy limit on the number of connections that may be associated with a particular client connection policy. We have to define at least two client connection policies, one for each of the two clients. Each policy must have different connection criteria for selecting the policy with which a given client connection should be associated.

Because the criteria is based on authentication, we must create a third client connection policy that applies to unauthenticated clients, because client connections are always unauthenticated as soon as they are established and before they have sent a bind request. Plus, clients are not required to send a bind request as their first operation.

Therefore, we define the following three client connection policies:

Client 1 Connection Policy, which only allows client 1, with an evaluation order index of 1.
Client 2 Connection Policy, which only allows client 2, with an evaluation order index of 2.
Unauthenticated Connection Policy, which allows unauthenticated clients, with an evaluation order index of 3.

We define simple connection criteria for the Client 1 Connection Policy and the Client 2 Connection Policy with the following properties:

The `user-auth-type` must not include none, so that it will only apply to authenticated client connections.
The `included-user-base-dn` should match the bind DN for the target user. This DN may be full DN for the target user, or it may be the base DN for a branch that contains a number of users that you want treated in the same way.

To create more generic criteria that match more than one user, you could list the DNs of each of the users explicitly in the `included-user-base-dn` property. If there is a group that contains all of the pertinent users, then you could instead use the `[all|any|not-all|not-any]-included-user-group-dn` property to apply to all members of that group. If the entries for all of the users match a particular filter, then you could use the `[all|any|not-all|not-any]-included-user-filter` property to match them.

How the Policy is Evaluated

Whenever a connection is established, the server associates the connection with exactly one client connection policy. The server does this by iterating over all of the defined client connection policies in ascending order of the evaluation order index. Policies with a lower evaluation order index value will be examined before those with a higher evaluation order index value. The first policy that the server finds whose criteria match the client connection will be associated with that connection. If no client connection policy is found with criteria matching the connection, then the connection will be terminated.

So, in our example, when a new connection is established, the server first checks the connection criteria associated with the Client 1 Connection Policy because it has the lowest evaluation order index value. If it finds that the criteria do not match the new connection, the server then checks the connection criteria associated with the Client 2 Connection Policy because it has the second lowest evaluation order index. If these criteria do not match, the server finally checks the connection criteria associated with the Unauthenticated Connection Policy, because it has the third lowest evaluation order index. It finds a match, so the client connection is associated with the Unauthenticated Connection Policy.

After the client performs a bind operation to authenticate to the server, then the client connection policies will be re-evaluated. If client 2 performs the bind, then the Client 1 Connection Policy will not match but the Client 2 Connection Policy will, so the connection will be re-associated with that client connection policy. Whenever a connection is associated with a client connection policy, the server will check to see if the maximum number of client connections have already been associated with that policy. If so, then the newly-associated connection will be terminated.

For example, Client 1 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. Client 1 then sends a bind request. The determination of whether the bind operation is allowed is made based on the constraints defined in the Unauthenticated Connection Policy, because it is the client connection policy already assigned to the client connection. Once the bind has completed, then the server will reevaluate the client connection policy against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to the Client 1 Connection Policy.

Next, Client 2 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. When Client 2 sends a bind request, the operation is allowed based on the constraints defined in the Unauthenticated Connection Policy. Once the bind is complete, the client connection is evaluated against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria do not match, so the client 2 connection is evaluated against the connection criteria associated with Client 2 Connection Policy, because it has the next lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to Client 2 Connection Policy.

Client 1 sends a search request. The Client 1 Connection Policy is used to determine whether the search operation should be allowed, because this is the client connection policy assigned to the client connection for client 1. The connection is not reevaluated, before or after processing the search operation.

To Configure a Client Connection Policy Using dsconfig

1. Use the `dsconfig` tool to create and configure a client connection policy.

   ```
   $ bin/dsconfig
   ```
2. Enter the connection parameters to the server (for example, hostname, connection method, port, bind DN and bind DN password).
3. In the Directory Proxy Server main menu, enter the number associated with client connection policy configuration. Then enter the number to create a new client connection policy.

   ```
   >>>> Client connection policy menu

   What would you like to do?

     1) List existing client connection policies
     2) Create a new client connection policy
     3) View and edit an existing client connection policy
     4) Delete an existing client connection policy
   ```

```
  b) back
  q) quit

Enter choice [b]: 2
```

4. Enter `n` to create a new client connection policy from scratch.

```
>>>> Select an existing Client Connection Policy to use as a
template for the new Client Connection Policy configuration or
‘n’ to create one from scratch:

  1) default

  n) new Client Connection Policy created from scratch
  c) cancel
  q) quit
```

5. Enter a name for the new client connection policy.

```
Enter the 'policy-id' for the Client Connection Policy that you

want to create: new_policy
```

6. Indicate whether you want the policy to be enabled by default.

```
Select a value for the 'enabled' property:

  1) true
  2) false

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

7. Provide a value for the `evaluation-order-index` property. Client connection policies with a lower index will be evaluated before those with a higher index.

```
Enter a value for the 'evaluation-order-index' property: 2
```

8. The properties of your new client connection policy are displayed. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the client connection policy.

   Any changes that you make to the client connection policy do not apply to existing connections. They will only apply to new connections.

Configuring Globally Unique Attributes

The PingDirectoryProxy Server supports a Globally Unique Attributes feature that ensures uniqueness for any value defined for a set of attributes within a subtree view. You can also configure when the server checks for attribute conflicts, either prior to any add, modify, or modify DN change request (pre-commit) or after the successful completion of a change request (post-commit).

About the Globally Unique Attribute Plug-in

The Directory Proxy Server supports a Globally Unique Attribute Plug-in that prevents any value within a defined set of attributes to appear more than once in any entry for one or more subtree views. Administrators can also configure whether conflict validation should be checked before an add, modify, or modify DN request to one or more backend servers or after the change has successfully completed.

For example, if the `pre-commit-validation` property is enabled, the Globally Unique Attribute Plugin performs one or more searches to determine whether any entries conflict with the change (i.e., add, modify, or modify DN). If a conflict is detected, then the change request will be rejected. If the `post-commit-validation` property is enabled, after the change has been processed, the server performs one or more searches to determine if a conflict was created in multiple servers at the same time. If a conflict is detected in this manner, then an administrative alert will be generated to notify administrators of the problem so that they can take any manual corrective action.

Note: The Globally Unique Attribute plug-in will attempt to detect and/or prevent unique attribute conflicts for changes processed through this Directory Proxy Server, but it cannot detect conflicts introduced by changes applied by clients communicating *directly* with backend servers.

We recommend that the Unique Attribute plug-in be enabled for all backend servers with the same configuration, so that conflicts can be detected within individual backend server instances. However, the Unique Attribute plug-in alone may not be sufficient for cases in which the content is split across multiple sets of servers (e.g., in an entry-balanced environment or in proxy configurations with different branches on different servers).

The LDAP SDK uniqueness request control can be used for enforcing uniqueness on a per-request basis. See the LDAP SDK documentation and the `com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl` class for using the control. See the ASN.1 specification to implement support for it in other APIs.

In general, note the following points about pre-commit validation versus post-commit validation:

- Pre-commit validation is the only mechanism that can try to prevent conflicts. It will increase the processing time for add, modify, and modify DN operations because the necessary searches to look for conflicts happen before the update request is forwarded to any backend servers.
- Post-commit validation will only let you know (via administrative alert) about conflicts that already exist in the data. It can't prevent conflicts, but can allow you to deal with them in a timely manner. It also operates during the post-response phase, so it won't affect the processing time for the associated write operation.
- In most cases, pre-commit validation should be sufficient to prevent conflicts, although we recommend that you periodically run the `identify-unique-attribute-conflicts` tool to find any conflicts that may have arisen. If you want to mitigate any risks due to conflicts being generated by concurrent operations in different servers, then using both `pre-commit-validation` and `post-commit-validation` properties provides the best combination of preventing most conflicts in advance, and detecting and alerting about conflicts that arise from concurrent writes.

For more detailed information about the plug-in, see the *Directory Proxy Server Reference (HTML)*

To Configure the Globally Unique Attribute Plug-in

The following example shows how to configure the Globally Unique Attribute plug-in. The example defines an attribute set consisting of the `telephoneNumber` and `mobile` attributes within the "test-view" subtree view. The `multiple-attribute-behavior` property determines the scope of how attributes may differ among entries and is the same property for the Directory Server plug-in. The property is set to `unique-across-all-attributes-including-in-same-entry`, which indicates that the `telephone` and `mobile` attributes must be unique throughout the subtree view, even within an entry. The `pre-commit-validation` property ensures that the Globally Unique Attribute Plugin performs one or more searches to determine whether any entries conflict with the change (i.e., add, modify, or modify DN). If a conflict is detected, then the change request will be rejected.

Note that all configured attributes should be indexed for equality in all backend servers.

- Run `dsconfig` to create the Globally Unique Attribute plug-in. The server will check that any add, modify, or modify DN request does not conflict with any attribute values in the entries. If there is a conflict, the change request will be rejected.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Globally-Unique telephone and mobile" \
  --type globally-unique-attribute \
  --set enabled:true \
  --set type:telephoneNumber \
  --set type:mobile \
  --set subtree-view:test-view \
  --set multiple-attribute-behavior:unique-across-all-attributes-including-
in-same-entry \
  --set pre-commit-validation:all-available-backend-servers
```

Configuring the Global Referential Integrity Plug-in

The PingDirectoryProxy Server supports a global referential integrity plug-in mechanism that maintains DN references from a specified set of attributes to entries that exist in the server (e.g., between the members values of a static group and the corresponding user entries). The plug-in intercepts delete and modify DN operations and updates any references to the target entry. For a delete operation, any references to the target entry are removed. For modify DN operations, any references to the target entry are updated to reflect the new DN of the entry.

The plug-in is similar to the Directory Server Referential Integrity Plug-in but does not have an asynchronous mode. When enabled on the Directory Proxy Server, the client response will be delayed until the referential integrity processing is complete. For Directory Proxy Server deployments not using entry balancing and using Directory Server external servers, it is best to instead use the Referential Integrity Plug-in on the Directory Server.

An equality index must be defined on all attributes referenced within the Global Referential Integrity Plug-in across all external servers.

Sample Global Referential Integrity Plug-in

- Use `dsconfig` to configure the Global Referential Integrity plug-in. The plug-in ensures that the `member`, `uniqueMember`, and `manager` attributes maintain their DN references in the defined subtree views. Note that any attributes for which referential integrity should be maintained should have values which are DNs and should be indexed for equality in all backend servers.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Global Referential Integrity" \
  --type global-referential-integrity \
  --set "enabled:true" \
  --set "attribute-type:member" \
  --set "attribute-type:uniqueMember" \
  --set "attribute-type:manager" \
  --set "subtree-view:employee-view" \
  --set "subtree-view:groups-view"
```

Configuring an Active Directory Server Back-end

Configuring an Active Directory server back-end requires a `dsconfig` script. The following settings are required for an Active Directory server:

- `verify-credentials-method:bind-on-existing-connections`, and `authorization-method:rebind`

  Active Directory does not support `proxy-as`. Existing connections must be reused.
- `set max-connection-age:5m`, and `health-check-pooled-connections:true`

  Active Directory drops idle connections after 15 minutes. The proxy needs to refresh the connection pool in a shorter interval.

The following example `dsconfig` script configures two Active Directory servers (AD-SRV1 and AD-SRV2).

```
dsconfig set-ldap-health-check-prop --check-name "Consume Admin Alerts" \
  --reset use-for-all-servers

dsconfig set-trust-manager-provider-prop \
  --provider-name "Blind Trust" \
  --set enabled:true

dsconfig create-external-server --server-name AD-SRV1 --type active-directory
 \
  --set server-host-name:example.server \
  --set server-port:636 \
  --set bind-dn:cn=ProxyUser,dc=dom-ad2,dc=local \
  --set password:password --set connection-security:ssl \
  --set key-manager-provider:Null --set trust-manager-provider:"Blind Trust" \
  --set authorization-method:rebind \
  --set verify-credentials-method:bind-on-existing-connections \
  --set max-connection-age:5m \
  --set health-check-pooled-connections:true

dsconfig create-external-server --server-name AD-SRV2 --type active-directory
 \
  --set server-host-name:example.server \
  --set server-port:636 \
  --set bind-dn:cn=ProxyUser,dc=dom-ad2,dc=local \
  --set password:password \
  --set connection-security:ssl \
  --set key-manager-provider:Null \
  --set trust-manager-provider:"Blind Trust" \
  --set authorization-method:rebind \
  --set verify-credentials-method:bind-on-existing-connections \
  --set max-connection-age:5m \
  --set health-check-pooled-connections:true

dsconfig create-load-balancing-algorithm --algorithm-name AD-LBA \
  --type fewest-operations \
  --set enabled:true \
  --set backend-server:AD-SRV1 \
  --set backend-server:AD-SRV2 \
  --set use-location:false

dsconfig create-request-processor --processor-name AD-Proxy --type proxying \
  --set load-balancing-algorithm:AD-LBA

dsconfig create-subtree-view --view-name AD-View \
  --set base-dn:dc=dom-ad2,dc=local \
  --set request-processor:AD-Proxy

dsconfig set-client-connection-policy-prop --policy-name default \
  --set subtree-view:AD-View
```

Chapter

4

Managing Access Control

Topics:

The PingDirectoryProxy Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and how it applies to the Directory Proxy Server.

Overview of Access Control

The access control model uses access control instructions (ACIs), which are stored in the aci operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and affects the entry or any subentries within that branch of the directory information tree (DIT).

Access control instructions specifies four items:

- **Resources**. *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the cn or userPassword password.
- **Name**. *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."
- **Clients**. *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."
- **Rights**. *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant read or write permission to a telephoneNumber attribute.

Key Access Control Features

The PingDirectoryProxy Server provides important access control features that provide added security for the Directory Proxy Server's entries.

Improved Validation and Security

The Directory Proxy Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Proxy Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed aci value will be rejected.

As an additional level of security, the Directory Proxy Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the lockdown-mode privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to an access control instruction not being enforced. When the problem has been corrected, the administrator can use the leave-lockdown-mode tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the server (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the PingDirectoryProxy Server, global ACIs are defined within the server configuration, in the global-aci property of configuration object for the access control handler. They can be viewed and managed using configuration tools like dsconfig and the Administrative Console.

The global ACIs available by default in the PingDirectoryProxy Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: namingContexts, subschemaSubentry, supportedAuthPasswordSchemes, supportedControl, supportedExtension, supportedFeatures, supportedLDAPVersion, supportedSASLMechanisms, vendorName, and vendorVersion.

- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

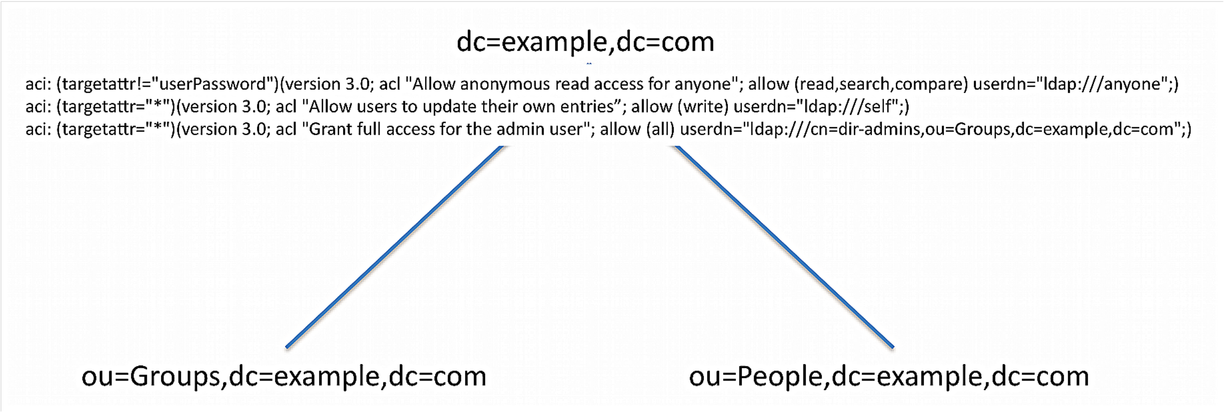
Access Controls for Public or Private Backends

The PingDirectoryProxy Server classifies backends as either public or private, depending on their intended purpose. A private backend is one whose content is generated by the Directory Proxy Server itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by the server (for example, the LDAP changelog backend). A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The PingDirectoryProxy Server access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General Format of the Access Control Rules

Access control instructions (ACIs) are represented as strings that are applied to one or more entries within the Directory Information Tree (DIT). Typically, an ACI is placed on a subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Directory Proxy Server iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissable operations. When a client application, such as `ldapsearch`, enters a request, the Directory Proxy Server checks that the user who binds with the server has the necessary access rights to the requested search targets. ACIs are cumulatively applied, so that a user who may have an ACI at an entry, may also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.



```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous read access for anyone"; allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")(version 3.0; acl "Allow users to update their own entries”; allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin user"; allow (all) userdn="ldap:///cn=dir-admins,ou=Groups,dc=example,dc=com";)
```

Figure 3: ACI

An access control rule has a basic syntax as follows:

```
aci : (targets) (version 3.0; acl "name"; permissions bind rules;)
```

Table 4: Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and/or attributes to which an access control rule applies. Syntax: *(target keyword = || != expression)*
name	Specifies the name of the ACI.
permissions	Specifies the type of operations to which an access control rule might apply. Syntax: *allow||deny (permission)*
bind rules	Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: *bind rule keyword = ||!= expression;*. The bind rule syntax requires that it be terminated with a ";".

Summary of Access Control Keywords

This section provides an overview of the keywords supported for use in the PingDirectoryProxy Server access control implementation.

Targets

A target expression specifies the set of entries and/or attributes to which an access control rule applies. The *keyword* specifies the type of target element. The *expression* specifies the items that is targeted by the access control rule. The operator is either the equal ("=") or not-equal ("!="). Note that the "!=" operator cannot be used with `targattrfilters` and `targetscope` keywords. For specific examples on each target keyword, see the section Working with Targets.

```
(keyword [=||!=]expression)
```

The following keywords are supported for use in the target portion of ACIs:

Table 5: Summary of Access Control Target Keywords

Target Keyword	Description	Wildcards
extop	Specifies the OIDs for any extended operations to which the access control rule should apply.	No
target	Specifies the set of entries, identified using LDAP URLs, to which the access control rule applies.	**Yes**
targattrfilters	Identifies specific attribute values based on filters that may be added to or removed from entries to which the access control rule applies.	**Yes**
targetattr	Specifies the set of attributes to which the access control rule should apply.	**Yes**
targetcontrol	Specifies the OIDs for any request controls to which the access control rule should apply.	No
targetfilter	Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.	**Yes**
targetscope	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI fi there is no target, to which the access control rule should apply.	No

Permissions

Permissions indicate the types of operations to which an access control rule might apply. You can specify if the user or group of users are allowed or not allowed to carry out a specific operation. For example, you would grant read access to a targeted entry or entries using "allow (read)" permission. Or you can specifically deny access to the target

entries and/or attributes using the "deny (read)" permission. You can list out multiple permissions as required in the ACI.

```
allow (permission1 ...,permission2,...permissionN)
```

```
deny (permission1 ...,permission2,...permissionN)
```

The following keywords are supported for use in the permissions portion of ACIs:

Table 6: Summary of Access Control Permissions

Permission	Description
add	Indicates that the access control should apply to add operations.
compare	Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
delete	Indicates that the access control should apply to delete operations.
export	Indicates that the access control should only apply to modify DN operations in which an entry is moved below a different parent by specifying a new superior DN in the modify DN request. The requestor must have the `export` permission for operations against the entry's original DN. The requestor must have the `import` permission for operations against the entry's new superior DN. For modify DN operations that merely alter the RDN of an entry but keeps it below the same parent (i.e., renames the entry), only the `write` permission is required. This is true regardless of whether the entry being renamed is a leaf entry or has subordinate entries.
import	See the description for the `export` permission.
proxy	Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identify to be specified).
read	Indicates that the access control rule should apply to search result entries returned by the server.
search	Indicates that the access control rule should apply to search operations with a non-base scope.
selfwrite	Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
write	Indicates that the access control rule should apply to modify and modify DN operations.
all	An aggregate permission that includes all other permissions except “proxy.” This is equivalent to providing a permission of “add, compare, delete, read, search, selfwrite, write, export, and import.”

Bind Rules

The Bind Rules indicate whether an access control rule should apply to a given requester. The syntax for the target keyword is shown below. The *keyword* specifies the type of target element. The expression specifies the items that is targeted by the access control rule. The operator is either equals ("=") or not-equals ("!="). The semi-colon delimiter symbol (";") is required after the end of the final bind rule.

```
keyword [=||!= ] expression;
```

Multiple bind rules can be combined using boolean operations (AND, OR, NOT) for more access control precision. The standard Boolean rules for evaluation apply: innermost to outer parentheses first, left to right expressions, NOT before AND or OR. For example, an ACI that includes the following bind rule targets all users who are not `uid=admin,dc=example,dc=com` and use simple authentication.

```
(userdn!="ldap:///uid=admin,dc=example,dc=com" and authmethod="simple");
```

The following bind rule targets the `uid=admin,dc=example,dc=com` and authenticates using SASL EXTERNAL or accesses the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or
 ip="127.0.0.1"));
```

The following keywords are supported for use in the bind rule portion of ACIs:

Table 7: Summary of Bind Rule Keywords

Bind Rule Keyword	Description

authmethod	Indicates that the requester's authentication method should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	`authmethod = method`
	where *method* is one of the following representations:
	none
	simple. Indicates that the client is authenticated to the server using a bind DN and password.
	ssl. Indicates that the client is authenticated with an SSL/TLS certificate (e.g., via SASL EXTERNAL), and not just over a secure connection to the server.
	sasl {sasl_mechanism}. Indicates that the client is authenticated to the server using a specified SASL Mechanism.
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (userdn="ldap:///self" and authmethod="ssl");)`

dayofweek	Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. Multiple day of week values may be separated by commas. The keyword's syntax is as follows:
	`dayofweek = day1, day2, ...`
	where *day* is one of the following representations:
	sun
	mon
	tues
	wed
	thu
	fri
	sat
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) on weekdays to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`

Bind Rule Keyword	Description

	`allow (write) (dayofweek!="sun,sat" and userdn="ldap:///self"`
	`and authmethod="ssl");)`

dns	Indicates that the requester's DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple DNS patterns may be separated by commas. The keyword's syntax is as follows:
	`dns` `=` *`dns-host-name`*
	The following example allows users on hostname `server.example.com` to update their own entries:
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write) (dns="server.example.com" and userdn="ldap:///self");)`

groupdn	Indicates that the requester's group membership should be taken into account when determining whether the access control rule should apply to any operation. Wildcards are not allowed in this expression.
	`groupdn [ = || != ] "ldap:///groupdn [ || ldap:///groupdn ] ..."`
	The following example allows users in the managers group to update their own entries:
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write)`
	`(groupdn="ldap:///cn=managers,ou=groups,dc=example,dc=com");)`

ip	Indicates that the requester's IP address should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple IP address patterns may be separated by commas. The keyword's syntax is as follows:
	`ip` `[ = || != ]` *`ipAddressList`*
	where *ipAddressList* is one of the following representations:
	A specific IPv4 address: 127.0.0.1
	An IPv4 address with wildcards to specify a subnetwork: 127.0.0.*
	An IPv4 address or subnetwork with subnetwork mask: 123.4.5.0+255.255.255.0
	An IPv4 address range using CIDR notation: 123.4.5.0/24
	An IPv6 address as defined by RFC 2373.
	The following example allows users on 10.130.10.2 and localhost to update their own entries:
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write) (ip="10.130.10.2,127.0.0.1" and userdn="ldap:///self");)`

<table>
<tr><th>Bind Rule Keyword</th><th>Description</th></tr>
<tr><td>timeofday</td><td>
Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:

<pre>
timeofday [= || != || >= || > || <= || <] time
</pre>

where *time* is one of the following representations:

4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour)
Wildcards are not allowed in this expression

The following example allows users to update their own entries if the request is received before 12 noon.

<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow users who authenticate before noon
    to update their own entries";
    allow (write) (timeofday<1200 and userdn="ldap:///self"
    and authmethod="simple");)
</pre>
</td></tr>
<tr><td>userattr</td><td>
Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of `USERDN` indicates that the target attribute should have a value which matches the DN of the authenticated user. A bindType value of `GROUPDN` indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A bindType value of `LDAPURL` indicates that the target attribute should have a value that is an LDAP URL whose criteria matches the entry for the authenticated user. Any value other than `USERDN`, `GROUPDN`, or `LDAPURL` is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:

<pre>
userattr = attrName# [bindType || attrValue]
</pre>

where:

attrName = name of the attribute for matching
bindType = USERDN, GROUPDN, LDAPURL
attrValue = an attribute value. Note that the *attrVALUE* of the attribute must match on both the bind entry and the target of the ACI.

The following example allows a manager to change employee's entries. If the bind DN is specified in the *manager* attribute of the targeted entry, the bind rule is evaluated to TRUE.

<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow a manager to change employee
 entries";
   allow (write) userattr="manager#USERDN";)
</pre>

The following example allows any member of a group to change employee's entries. If the bind DN is a member of the group specified in the *allowEditors* attribute of the targeted entry, the bind rule is evaluated to TRUE.

<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow allowEditors to change employee
 entries";
   allow (write) userattr="allowEditors#GROUPDN";)
</pre>
</td></tr>
</table>

<table>
<tr><th>Bind Rule Keyword</th><th>Description</th></tr>
<tr><td></td><td>
The following example allows allows a user's manager to edit that user's entry and any entries below the user's entry up to two levels deep. You can specify up to five levels (0, 1, 2, 3, 4) below the targeted entry, with zero (0) indicating the targeted entry.

<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow managers to change employees entries
 two levels below";
    allow (write) userattr="parent[0,1,2].manager#USERDN";)
</pre>

The following example allows any member of the engineering department to update any other member of the engineering department at or below the specified ACI.

<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow any member of Eng Dept to update any
 other member of the
   enginering department at or below the ACI";
   allow (write) userattr="department#ENGINEERING";)
</pre>

The following example allows an entry to be updated by any user whose entry matches the criteria defined in the LDAP URL contained in the `allowedEditorCriteria` attribute of the target entry.

<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow a user that matches the filter to
 change entries";
   allow (write) userattr="allowedEditorCriteria#LDAPURL";)
</pre>
</td></tr>
<tr><td>userdn</td><td>
Indicates that the user’s DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword’s syntax is as follows:

<pre>
userdn [= || !=] "ldap:///value [|| "ldap:///value ..."]
</pre>

where *value* is one of the following representations:

The DN of the target user
A value of `anyone` to match any client, including unauthenticated clients.
A value of `all` to match any authenticated client.
A value of `parent` to match the client authenticated as the user defined in the immediate parent of the target entry.
A value of `self` to match the client authenticated as the user defined in the target entry.

If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.

The following example allows users to update their own entries:

<pre>
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
</pre>
</td></tr>
</table>

Working with Targets

The following section presents a detailed look and examples of the target ACI keywords: `target`, `targetattr`, `targetfilter`, `targattrfilters`, `targetscope`, `targetcontrol`, and `extop`.

target

The `target` keyword indicates that the ACI should apply to one or more entries at or below the specified distinguished name (DN). The target DN must be equal or subordinate to the DN of the entry in which the ACI is placed. For example, if you place the ACI at the root of `ou=People,dc=example,dc=com`, you can target the DN, `uid=user.1,ou=People,dc=example,dc=com` within your ACI rule. The DN must meet the string representation specification of distinguished names, outlined in RFC 4514, and requires that special characters be properly escaped.

The `target` clause has the following format, where DN is the distinguished name of the entry or branch:

```
(target = ldap:///DN)
```

For example, to target a specific entry, you would use a clause such as the following:

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

Note that, in general, specifying a target DN is not recommended. It is better to have the ACI defined in that entry and omit the `target` element altogether. For example, although you can have `(target="ldap:///uid=john.doe,ou=People,dc=example,dc=com)` in any of the `dc=example,dc=com` or `ou=People` entries, it is better for it to be defined in the `uid=john.doe` entry and not explicitly include the `target` element.

The expression allows for the "not equal" (!=) operator to indicate that all entries within the scope of the given branch that do NOT match the expression be targeted for the ACI. Thus, the following expression targets all entries within the subtree that do not match `uid=john.doe`.

```
(target != ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

The `target` keyword also supports the use of asterisk (*) characters as wildcards to match elements within the distinguished name. The following target expression matches all entries that contains and begins with "john.d, " so that entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=People,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
```

The following target expression matches all entries whose DN begins with "john.d," and matches the `ou` attribute. Entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=asia-branch,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
```

Another example of a complete ACI targets the entries in the `ou=People,dc=example,dc=com` branch and the entries below it, and grants the users the privilege to modify all of their user attributes within their own entries.

```
aci:(target="ldap:///ou=People,dc=example,dc=com")
  (targetattr="*")
  (version 3.0; acl "Allow all the ou=People branch to modify their own
 entries";
  allow (write) userdn="ldap:///self";)
```

targetattr

The `targetattr` keyword targets the attributes for which the access control instruction should apply. There are four general forms that it can take in the PingDirectoryProxy Server:

- **(targetattr="*")**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **(targetattr="+")**. Indicates that the access control rule applies to all operational attributes. User attributes will not automatically be included in this set.
- **(targetattr="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies only to the named set of attributes.

- **(targetattr!="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as user attributes and operational attributes. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes, such as when the entry was created, last modified and by whom. Metadata can also include attributes specifying which password policy applies to the user, or overrided default constraints like size limit, time limit, or look-through limit for that user.

The PingDirectoryProxy Server distinguishes between these two types of attributes in its access control implementation. The Directory Proxy Server does not automatically grant any access at all to operational attributes. For example, the following clause applies only to user attributes and not to operational attributes:

```
(targetattr="*")
```

You can also target multiple attributes in the entry. The following clause targets the common name (cn), surname (sn) and state (st) attribute:

```
(targetattr="cn||sn||st")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

To include all user and all operational attributes, you use both symbols, as follows:

```
(targetattr="*||+")
```

If there is a need to target a specific operational attribute rather than all operational attributes, then it can be specifically included in the values of the `targetattr` clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or if you want to target all user attributes and a specific operational attribute, then you can use them in the `targetattr` clause, as follows:

```
(targetattr="*||ds-rlim-size-limit")
```

The following ACIs are placed on the `dc=example,dc=com` tree and allows any user anonymous read access to all entries except the `userPassword` attribute. The second ACI allows users to update their own contact information. The third ACI allows the `uid=admin` user full access privileges to all user attributes in the `dc=example,dc=com` subtree.

```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous
   read access for anyone"; allow (read,search,compare) userdn="ldap:///
anyone";)
aci: (targetattr="telephonenumber||street||homePhone||l||st")
  (version 3.0; acl "Allow users to update their own contact info";
   allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin
 user";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

An important note must be made when assigning access to user and operational attributes, which can be outlined in an example to show the implications of the Directory Proxy Server not distinguishing between these attributes. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of `uid` and `employeeNumber`. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a Directory Proxy Server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users including administrators granted full rights by access control rules, but excluding users with the `bypass-acl` privilege, allow them to edit any other attributes in their own entry including those excluded by rules like `uid` and `employeeNumber` in the example above, or add, modify, or delete any entries below his or her own entry.

Because the PingDirectoryProxy Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the `modify-acl` privilege.

Another danger in using the `(targetattr!="x")` pattern is that two ACIs within the same scope could have two different `targetattr` policies that cancel each other out. For example, if one ACI has `(targetattr!="cn||sn")` and a second ACI has `(targetattr!="userPassword")`, then the net effect is `(targetattr="*")`, because the first ACI inherently allows `userPassword`, and the second allows `cn` and `sn`.

targetfilter

The `targetfilter` keyword targets all attributes that match results returned from a filter. The `targetfilter` clause has the following syntax:

```
(targetfilter = ldap_filter)
```

For example, the following clause targets all entries that contain "ou=engineering" attribue:

```
(targetfilter = "(ou=engineering)")
```

You can only specify a single filter, but that filter can contain multiple elements combined with the OR operator. The following clause targets all entries that contain "ou=engineering," "ou=accounting," and "ou=marketing."

```
(targetfilter = "(|(ou=engineering)(ou=accounting)(ou=marketing)")
```

The following example allows the user, `uid=eng-mgr`, to modify the `departmentNumber`, `cn`, and `sn` attributes for all entries that match the filter `ou=engineering`.

```
aci:(targetfilter="(ou=engineering)")
  (targetattr="departmentNumber||cn||sn")
  (version 3.0; acl "example"; allow (write)
   userdn="ldap:///uid=eng-mgr,dc=example,dc=com";)
```

targattrfilters

The `targattrfilters` keyword targets specific attribute *values* that match a filtered search criteria. This keyword allows you to set up an ACI that grants or denies permissions on an attribute value if that value meets the filter criteria. The `targattrfilters` keyword applies to individual values of an attribute, not to the whole attribute. The keyword also allows the use of wildcards in the filters.

The keyword clause has the following formats:

```
(target = "add=attr1:Filter1 && attr2:Filter2... && attrn:FilterN,
del=attr1:Filter1 && attr2:Filter2 ... && attrN:FilterN" )
```

where

add represents the operation of adding an attribute value to the entry

del represents the operation of removing an attribute value from the entry

attr1, attr2... attrN represents the targeted attributes

filter1, filter2 ... filterN represents filters that identify matching attribute values

The following conditions determine when the attribute must satisfy the filter:

- When adding or deleting an entry containing an attribute targeted a `targattrfilters` element, each value of that attribute must satisfy the corresponding filter.
- When modifying an entry, if the operation adds one or more values for an attribute targeted by a `targattrfilters` element, each value must satisfy the corresponding filter. If the operation deletes one or more values for a targeted attribute, each value must satisfy the corresponding filter.
- When replacing the set of values for an attribute targeted by a `targattrfilters` element, each value removed must satisfy the delete filters, and each value added must satisfy the add filters.

The following example allows any user who is part of the `cn=directory server admins` group to add the `soft-delete-read` privilege.

```
aci:(targattrfilter="add=ds-privilege-name:(ds-privilege-name=soft-delete-
read)")
  (version 3.0; acl "Allow members of the directory server admins group to
 grant the
   soft-delete-read privilege"; allow (write)
   groupdn="ldap:///cn=directory server admins,ou=group,dc=example,dc=com";)
```

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is base to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
    (targetattr="supportedControl||supportedExtension||
     supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
    (version 3.0; acl "Allow users to view Root DSE Operational Attributes";
     allow (read,search,compare) userdn="ldap:///anyone")
```

targetcontrol

The `targetcontrol` keyword is used to indicate whether a given request control can be used by those users targeted in the ACI. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying control OIDs.

The following ACI example shows the controls required to allow an administrator to use and manage the Soft-Delete feature. The Soft Delete Request Control allows the user to soft-delete an entry, so that it could be undeleted at a later time. The Hard Delete Request Control allows the user to permanently remove an entry or soft-deleted entry. The Undelete Request Control allows the user to undelete a currently soft-deleted entry. The Soft-Deleted Entry Access Request Control allows the user to search for any soft-deleted entries in the server.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.22||
   1.3.6.1.4.1.30221.2.5.23||1.3.6.1.4.1.30221.2.5.24")
  (version 3.0; acl "Allow admins to use the Soft Delete Request Control,
   Hard Delete Request Control,Undelete Request Control, and
   Soft-deleted entry access request control";
   allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

extOp

The `extop` keyword can be used to indicate whether a given extended request operation can be used. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying extended request OIDs.

The following ACI allows the `uid=user-mgr` to use the Password Modify Request (i.e., OID=1.3.6.1.4.1.4203.1.11.1) and the StartTLS (i.e., OID=1.3.6.1.4.1.1466.20037) extended request OIDs.

```
aci:(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")
  (version 3.0; acl "Allows the mgr to use the Password Modify Request and
 StartTLS;
   allow(read) userdn="ldap:///uid=user-mgr,ou=people,dc=example,dc=com";)
```

Examples of Common Access Control Rules

This section provides a set of examples that demonstrate access controls that are commonly used in your environment. Note that to be able to alter access control definitions in the server, a user must have the `modify-acl` privilege as discussed later in this chapter.

Administrator Access

The following ACI can be used to grant any member of the "`cn=admins,ou=groups,dc=example,dc=com`" group to add, modify and delete entries, reset passwords and read operational attributes such as `isMemberOf` and password policy state:

```
aci: (targetattr="+")(version 3.0; acl "Administrators can read, search or
 compare operational attributes";
allow (read,search,compare) groupdn="ldap:///
cn=admins,ou=groups,dc=example,dc=com";)
aci: (targetattr="*")(version 3.0; acl "Administrators can add, modify and
 delete entries";
allow (all) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

Anonymous and Authenticated Access

The following ACI allow anonymous read, search and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. The authenticated user will inherit the privileges of the anonymous ACI. In addition, the authenticated user can change `userPassword`:

```
aci: (targetattr="objectclass || uid || cn || mail || sn || givenName")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Anyone can access names and email addresses of entries
 representing people";
allow (read,search,compare) userdn="ldap:///anyone";)
```

```
aci: (targetattr="departmentNumber || manager || isMemberOf")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Authenticated users can access these fields for entries
 representing people";
allow (read,search,compare) userdn="ldap:///all";)
aci: (targetattr="userPassword")(version 3.0; acl "Authenticated users can
 change password";
allow (write) userdn="ldap:///all";)
```

If no unauthenticated access should be allowed to the Directory Server, the preferred method for preventing unauthenticated, or anonymous access is to set the Global Configuration property `reject-unauthenticated-requests` to true.

Delegated Access to a Manager

The following ACI can be used to allow an employee's manager to edit the value of the employee's `telephoneNumber` attribute. This ACI uses the `userattr` keyword with a bind type of `USERDN`, which indicates that the target entry's manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
(version 3.0; acl "A manager can update telephone numbers of her direct
 reports";
allow (read,search,compare,write) userattr="manager#USERDN";)
```

Proxy Authorization

The following ACIs can be used to allow the application "`cn=OnBehalf,ou=applications,dc=example,dc=com`" to use the proxied authorization v2 control to request that operations be performed using an alternate authorization identity. The application user is also required to have the `proxied-auth` privilege as discussed later in this chapter:

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com";)
```

Validating ACIs Before Migrating Data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Sun/Oracle encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the PingDirectoryProxy Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a PingDirectoryProxy Server.

To validate an access control instruction, the PingDirectoryProxy Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values, and all other attributes will be ignored.

To Validate ACIs from a File

Content-Reference to:

The `validate-acis` tool can process data contained in an LDIF file. It will ignore all attributes except `aci`, and will ignore all entries that do not contain the `aci` attribute, so any existing LDIF file that contains access control rules may be used.

1. Run the `bin/validate-acis` tool (UNIX or Linux systems) or `bat\validate-acis` (Win dows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

   ```
   $ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-
   acis.ldif
   ```

   ```
   # Processing complete # Total entries examined: 1
   # Entries found with ACIs: 1
   # Total ACI values found: 3
   # Malformed ACI values found: 0
   # Other processing errors encountered: 0
   ```
2. Review the results by opening the output file. For example, the `validated-acis.ldif` file that was generated in the previous step reads as follows:

   ```
   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr!="userPassword")
     (version 3.0; acl "Allow anonymous read access for anyone";
       allow (read,search,compare) userdn="ldap:///anyone";)

   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr="*")
     (version 3.0; acl "Allow users to update their own entries";
       allow (write) userdn="ldap:///self";)

   # The following access control rule is valid
   dn: dc=example,dc=com
   aci: (targetattr="*")
     (version 3.0; acl "Grant full access for the admin user";
       allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
   ```
3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed ACI syntax is encountered.

   ```
   $ bin/validate-acis --ldifFIle test-acis.ldif --outputFile validated-
   acis.ldif \
     --onlyReportErrors
   ```

   ```
   # Processing complete
   # Total entries examined: 1
   # Entries found with ACIs: 1
   # Total ACI values found: 3
   # Malformed ACI values found: 0
   # Other processing errors encountered: 0
   ```

   The output file shows the following message:

   ```
   # The following access control rule is malformed or contains an unsupported
   # syntax: The provided string '(targetattr!=userPassword)(version 3.0; acl
   # "Allow anonymous read access for anyone"; allow (read,search,compare)
   # userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
   # Instruction (ACI) because it failed general ACI syntax evaluation
   dn: dc=example,dc=com
   aci: (targetattr!=userPassword)
     (version 3.0; acl "Allow anonymous read access for anyone";
       allow (read,search,compare) userdn="ldap:///anyone";)

   # The following access control rule is valid
   ```

```
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Validate ACIs in Another Directory Proxy Server

The `validate-acis` tool also provides the ability to examine ACIs in data that exists in another Directory Proxy Server that you are planning to migrate to the PingDirectoryProxy Server. The tool helps to determine whether the Ping Identity Server accepts those ACIs.

- To use it in this manner, provide arguments that specify the address and port of the target Directory Proxy Server, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

Migrating ACIs from Sun/Oracle to PingDirectory Server

This section describes the most important differences in access control evaluation between Sun/Oracle and the PingDirectory Server.

Support for Macro ACIs

Sun/Oracle provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the PingDirectory Server. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN Bind Rule

Sun/Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The PingDirectory Server does not support DSEE roles and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting Operational Attributes

The Sun/Oracle access control model does not differentiate between user attributes and operational attributes. With Sun/Oracle, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible for several significant security holes in which users are unintentionally given access to operational attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the PingDirectory Server, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except

`userPassword`, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like `targetattr="creatorsName||modifiersName"`. All operational attributes can be targeted using the `"+"` character. So, `targetattr="+"` targets all operational attributes but no user attributes and `targetattr="*||+"` targets all user and operational attributes.

Specification of Global ACIs

Both DSEE and PingDirectory Server support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the `aci` attribute. In the PingDirectory Server, global ACIs are managed with `dsconfig` referenced in the `global-aci` property of the Access Control Handler.

Defining ACIs for Non-User Content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the PingDirectory Server, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting Access to Controls and Extended Operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as "`oid={oid},cn=features,cn=config`" where `{oid}` is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control: "`oid=2.16.840.1.113730.3.4.3,cn=features,cn=config`".

In the PingDirectory Server, the "`targetcontrol`" keyword can be used to define ACIs that grant or deny access to controls. The "`extop`" keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for Malformed ACI Values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The PingDirectory Server is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the `lockdown-mode` privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.

Note: Consider running the `import-ldif` tool with the --rejectFile option so that you can review any rejected ACIs.

About the Privilege Subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you cannot restrict the capabilities assigned to that root user.

The PingDirectory Server offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server

configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying Unsupported ACIs

The PingDirectory Server provides a `validate-acis` tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the PingDirectory Server instance. When migrating data from a DSEE deployment into a PingDirectory Server instance, the `validate-acis` tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the PingDirectory Server.

For more information about using this tool, see Validating ACIs Before Migrating Data.

Working with Privileges

In addition to the access control implementation, the PingDirectoryProxy Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the PingDirectoryProxy Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available Privileges

The following privileges are defined in the PingDirectoryProxy Server.

Table 8: Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the `audit-data-security` tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the `bypass-acl` privilege but without the `config-read` privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is not evaluated for bind

Privilege	Description
	operations so that password policy evaluation will still occur when binding as a user with this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the `config-read` privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the Directory Proxy Server. The user is also required to have the `jmx-read` privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Proxy Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Proxy Server via the Java Management Extensions (JMX). The user is also required to have the `jmx-read` privilege. Note that currently all of the information exposed by the server over JMX is read-only.
ldif-export	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group.
ldif-import	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the `ldif-export` privilege.
lockdown-mode	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.
modify-acl	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.
password-reset	This privilege is required for one user to be allowed to change another user’s password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.
privilege-change	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the `ds-privilege-name` operational attribute.

Privilege	Description
proxied-auth	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
server-restart	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
server-shutdown	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
soft-delete-read	This privilege is required for a user to access a soft-deleted-entry.
stream-values	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
unindexed-search	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
update-schema	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges Automatically Granted to Root Users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the `ds-cfg-inherit-default-root-privileges` attribute with a value of FALSE in the entries for those root users.
- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the `default-root-privilege-name` property of the Root DN configuration object. By default, this set of privileges includes:

audit-data-security
backend-backup
backend-restore
bypass-acl
config-read
config-write
disconnect-client
ldif-export
lockdown-mode
manage-topology
metrics-read
modify-acl
password-reset

permit-get-password-policy-state-issues
privilege-change
server-restart
server-shutdown
soft-delete-read
stream-values
unindexed-search
update-schema

The privileges not granted to root users by default includes:

bypass-pw-policy
bypass-read-acl
jmx-read
jmx-write
jmx-notify
permit-externally-processed-authentication
permit-proxied-mschapv2-details
proxied-auth

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the `config-read`, `config-write`, and `privilege-change` privileges, as well as either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning Additional Privileges for Administrators

To allow access to the Tasks backend, set up a global ACI that allows access to members of an Administrators group as follows:

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*||+")
        (version 5.0; acl "Access to the tasks backend for administrators";
         allow (all) groupdn="ldap:///
         cn=admins,ou=groups,dc=example,dc=com";)'
```

Assigning Privileges to Normal Users and Individual Root Users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling Privileges

Although the privilege subsystem in the PingDirectoryProxy Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Chapter 5

Deploying a Standard Directory Proxy Server

Topics:

You can deploy PingDirectoryProxy Server in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates a standard deployment scenario.

Creating a Standard Multi-Location Deployment

In this example deployment, PingDirectoryProxy Server will be deployed in the data centers of two geographic locations: east and west. All LDAP external servers in this deployment are PingDirectory Servers. The directory servers in the eastern city are assigned to the location named east, and the directory servers in the western city are assigned to the location named west.

This example refers to four PingDirectory Server instances in two locations with replication of the `dc=example,dc=com` base DN enabled:

ds-east-01.example.com
ds-east-02.example.com
ds-west-01.example.com
ds-west-01.example.com

We will configure four Directory Proxy Server instances:

proxy-east-01.example.com
proxy-east-02.example.com
proxy-west-01.example.com
proxy-west-02.example.com

Overview of the Deployment Steps

In this deployment scenario, we will take the following steps:

- Install the first Directory Proxy Server in east location using the `setup` or `setup.bat` file included in the zip installation file.
- Use the `create-initial-proxy-config` tool to provide a proxy user bind DN and password, define locations for each of our data centers, and configure the LDAP external servers in these data centers.
- Test external server communications after initial setup is complete and test a simulated external server failure.
- Install the second proxy server in the east location using the `setup` or `setup.bat` file included in the zip installation file and copy the configuration of the first Directory Proxy Server using the configuration cloning feature.
- Install two Directory Proxy Server instances in the west location, which includes using the setup file and manually setting the location to west using the `dsconfig` command, as well as copying the configuration of the Directory Proxy Server using the configuration cloning feature.

After the proxy server has been configured and tested, we then provide a tour of the configuration of each of the proxy server components. These properties can be modified later as needed using the `dsconfig` tool.

Installing the First Directory Proxy Server

To begin with, we have the PingDirectoryProxy Server installation zip file. In this example, we plan to use SSL security, so we also have a keystore certificate database and a pin file that contains the private key password for the keystore. The keystore files are only necessary when using SSL or StartTLS.

In this deployment scenario, the keystore database is assumed to be a Java Keystore (JKS), which can be created by the keytool program. For more information about using the keytool, see the "Security Chapter" in the PingDirectory Server Administration Guide.

The PingDirectoryProxy directory contains the following:

```
root@proxy-east-01: ls
ExampleKeystore.jks   ExampleTruststore.jks ExampleKeystore.pin
PingDirectoryProxy-7.0.0.0-with-je.zip
```

The `ExampleKeystore.jks` keystore file contains the private key entry for the `proxy-east-01.example.com` server certificate with the alias `server-cert`. The server certificate, CA, and intermediate signing certificates are all contained in the `ExampleTruststore.jks` file. The password for `ExampleKeystore.jks` is defined in clear text in the corresponding pin file, though the name of the file need not match as it does in our example. The private key password in our example is the same as the password defined for the `ExampleKeystore.jks` keystore.

To Install the First Directory Proxy Server

1. Unzip the compressed archive file into the PingDirectoryProxy directory and move to this directory.

```
root@proxy-east-01: unzip -q PingDirectoryProxy-<version>-with-je.zip
root@proxy-east-01: cd PingDirectoryProxy
```

2. Because we are configuring SSL security, copy the keystore and pin files into the `config` directory.

```
root@proxy-east01: cp ../*Keystore* config/
root@proxy-east01: cp ../*Truststore* config/
```

3. Next, we install the first proxy server by running the `setup` tool on `proxy-east-01.example.com` as follows:

```
root@proxy-east01: ./setup --no-prompt --acceptLicense \
--ldapPort 389 --rootUserPassword pass \
--aggressiveJVMTuning --maxHeapSize 1g \
--enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickname server-cert \
--useJavaTrustStore config/ExampleTruststore.jks
```

New keystore password files are created in `config/keystore.pin`. The original file, `config/ExampleKeystore.pin`, is no longer needed.

4. If you are not using SSL or StartTLS, then the SSL arguments are not necessary as follows:

```
root@proxy-east01: ./setup --no-prompt --acceptLicense \
--ldapPort 389 --rootUserPassword pass \
--aggressiveJVMTuning --maxHeapSize 1g
```

Once you have installed the Directory Proxy Server, you can configure it using the `create-initial-proxy-config` tool as presented in the next section.

Configuring the First Directory Proxy Server

Once the Directory Proxy Server has been installed, it can be automatically configured using the `create-initial-proxy-config` tool. This tool can only be used once for this initial configuration, after which we will have to use `dsconfig` to make any changes to our proxy server configuration.

Configuring the Directory Proxy Server with the `create-initial-proxy-config` tool involves the following steps:

Providing a Directory Proxy Server base DN and password.
Defining locations for each of our data centers, east and west.
Configuring the LDAP external server in the east location.
Configuring the LDAP external servers in the west location.
Applying the changes to the Directory Proxy Server.

To Configure the First Directory Proxy Server

1. Once we have completed setup, we run the `create-initial-proxy-config` tool as follows:

```
root@proxy-east01: bin/create-initial-proxy-config
```

2. Provide the bind DN and password that the Directory Proxy Server will use to authenticate to the backend PingDirectory Server instances. The `create-initial-proxy-config` tool requires that the same bind

DN and password be used to authenticate to all of the backend servers. All Directory Proxy Server instances have identical proxy user accounts and passwords. If necessary, the proxy user account password can be defined differently for each external server using `dsconfig` after the `create-initial-proxy-config` tool has been executed.

3. Specify the type of external server communication security that will be used to communicate with the PingDirectory Server instances. For this example, enter the option for 'None'.
4. Specify the base DNs of the PingDirectory Server instances that the Directory Proxy Server will access. For this example, use `dc=example,dc=com`.
5. Enter any other base DNs of the PingDirectory Server instances that will be accessed through the proxy server. Because we are only using one proxy base DN, press **Enter** to finished.

Defining Locations

Next, we define our first location, east, to accommodate the servers in our deployment located on the East Coast of the United States.

To Define Proxy Locations

1. Continuing from the same `create-initial-proxy-config` session, enter a location name for the Directory Proxy Server. In this example, enter `east`, and then press **Enter**.
2. Define a location named `west` for the servers in our deployment located on the West Coast. Press **Enter** when finished.
3. Select the location that contains the Directory Proxy Server itself. The Directory Proxy Server is located in the east.

Configuring the External Servers in the East Location

Once the locations have been defined, we need to identify the directory servers. First, we define one of the servers in the east location.

To Configure the External Servers in the East Location

1. Define one of the servers in the east location by entering the host name and port of the server. For this example, enter `ds-east-01.example.com:389`.

```
>>>> External Servers

External Servers identify directory server instances including
host, port, and authentication information.

Enter the host and port (host:port) of the first directory server
in 'east'

    b)  back
    q)  quit

Enter a host:port or choose a menu item [localhost:389]: ds-
east-01.example.com:389
```

2. Enter the option to prepare the server and all subsequent servers. Preparing the servers involves testing the connections to these servers and sets up the `cn=Proxy User` account on the Directory Proxy Server.
3. Enter the DN of the account with which to manage the `cn=Proxy User,cn=Root DNs,cn=config` account. For this example, use the default, `cn=Directory Manager`.
4. Repeat the previous steps to prepare the other server in the east location, ds-east-02.example.com.
5. Press **Enter** to complete preparing the servers.

To Configure the External Servers in the West Location

The same process used for the east location is used to define the LDAP external servers for the west location.

1. Define the first external server, ds-west-01.example.com.
2. Define the second server in the west location, ds-west-02.example.com.
3. Press **Enter** when finished.

Apply the Configuration to the Directory Proxy Server

Next, we review the configuration summary. Once we have confirmed that the changes are correct, we press **Enter** to write the configuration.

To Apply the Changes to the Directory Proxy Server

1. During the configuration process, the `create-initial-proxy-config` tool writes the configuration settings to a `dsconfig` batch file, which will then be applied to the Directory Proxy Server. The batch file can be reused to configure other servers. On the final step, the `create-initial-proxy-config` tool presents a configuration summary. Review the configuration and then apply the changes to the Directory Proxy Server. Press **Enter** to write the configuration to the server.
2. On the final confirmation prompt, press **Enter** to apply the changes to the proxy server, and then enter the LDAP connection parameters to the server. Once the changes have been applied, the `create-initial-proxy-config` tool cannot be used to configure this proxy server again.

Configuring Additional Directory Proxy Server Instances

We install and configure the second Directory Proxy Server by running the `setup` tool on proxy-east-02.example.com.

To Configure Additional Directory Proxy Server Instances

1. Copy the keystore and pin files into the `config` directory for the proxy-east-02.example.com server.

```
root@proxy-east-02: cp ../*Keystore* config/
root@proxy-east-02: cp ../*Truststore* config/
```

2. Install the second Directory Proxy Server by running the `setup` tool on proxy-east-02.example.com as follows:

```
root@proxy-east-02: ./setup --no-prompt \
--listenAddress proxy-east-02.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-east-02.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location east
```

3. Configure the third Directory Proxy Server, proxy-west-01.example.com in the same way as shown in the previous step. First, copy the keystore and pin files into the `config` directory.

```
root@proxy-west-01: cp ../*Keystore* config/
root@proxy-west-01: cp ../*Truststore* config/
```

4. Run the `setup` tool on proxy-west-01.example.com as follows:

```
root@proxy-west-01: ./setup --no-prompt \
--listenAddress proxy-west-01.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-west-01.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location west
```

5. Finally, repeat steps 3 and 4 to install the last Directory Proxy Server by first copying the keystore and pin files to the `config` directory and then running the `setup` command.

   At this point, all proxies have the same Admin Data backend and have the `all-servers` group defined as their configuration-server-group in the Directory Proxy Server Global Configuration object. When making a change to a Directory Proxy Server using the `dsconfig` command-line tool or the Administrative Console, you will have the choice to apply the changes locally only or to all proxies in the `all-servers` group.

Testing External Server Communications After Initial Setup

After setting up the basic deployment scenario, the communication between the proxies and the LDAP external servers can be tested using a feature in the proxy server in combination with an LDAP search.

To Test the External Communications After Initial Setup

After initial setup, the Directory Proxy Server exposes a special search base DN for testing external server connectivity, called the `backend server pass-through` subtree view. While disabled by default, you can enable this feature using `dsconfig` in the Client Connection Policy menu. Set the value of the `backend-server-passthrough-subtree-views` property to `TRUE`.

1. Run `dsconfig` to set the `include-backend-server-passthrough-subtree-views` property to `TRUE`.

```
root@proxy-east-01: dsconfig set-client-connection-policy-prop \
--policy-name default \
--set include-backend-server-passthrough-subtree-views:true
```

   Once set to true, an LDAP search against the Directory Proxy Server with the base DN `dc=example,dc=com,ds-backend-server=ds-east-02.example.com:389` instructs the Directory Proxy Server to perform the search against the `ds-east-02.example.com:389` external server with the base DN set to `dc=example,dc=com`. The value of `ds-backend-server` should be the name of the configuration object representing the external server. Depending on your naming scheme, this name may not be a `host:port` combination.

2. Run `ldapsearch` to fetch the `dc=example,dc=com` entry from the `ds-east-01.example.com` server. Perform this search on each external server to determine if external server communication has been configured correctly on the Directory Proxy Server.

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--baseDN "dc=example,dc=com,ds-backend-server=ds-east-01.example.com:389" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. You can also use this special subtree view to track the operations performed on each external server to help determine load balancing requirements. This LDAP search can be run with the base DN values for the `ds-east-01` and `ds-east-02` servers to track the distribution of search and bind requests over time. These statistics are reset to zero when the server restarts. The following example searches an external server's monitor entry to display operation statistics:

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=directory manager" \
--bindPassword password \
--baseDN "cn=monitor,ds-backend-server=ds-east-02.example.com:389" \
--searchScope sub --useStartTLS "(cn=ldap*statistics)"

dn: cn=LDAP Connection Handler 192.168.1.203 port 389
```

```
Statistics,cn=monitor,ds-backend-server=ds-east-02.example.com:389

objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-ldap-statistics-monitor-entry
objectClass: extensibleObject
cn: LDAP Connection Handler 192.168.1.203 port 389
Statistics
connectionsEstablished: 3004
connectionsClosed: 2990
bytesRead: 658483
bytesWritten: 2061549
ldapMessagesRead: 17278
ldapMessagesWritten: 22611
operationsAbandoned: 0
operationsInitiated: 17278
operationsCompleted: 14241
abandonRequests: 22
addRequests: 1
addResponses: 1
bindRequests: 3006
bindResponses: 3006
compareRequests: 0
compareResponses: 0
deleteRequests: 0
deleteResponses: 0
extendedRequests: 2987
extendedResponses: 2987
modifyRequests: 1
modifyResponses: 1
modifyDNRequests: 0
modifyDNResponses: 0
searchRequests: 8271
searchResultEntries: 8370
searchResultReferences: 0
searchResultsDone: 8246
unbindRequests: 2990
```

Testing a Simulated External Server Failure

Once you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the Directory Proxy Server redirects LDAP requests appropriately. In this procedure, we stop the ds-east-01.example.com:389 server instance and test searches through proxy-east-01.example.com.

To Test a Simulated External Server Failure

1. First, perform several searches against the Directory Proxy Server. Verify activity in each of the servers in the east location, ds-east-01 and ds-east-02, by looking at the access logs. Because we used the default load balancing algorithm of fewest operations, it is likely that all of the searches will go to only one of the proxies. The following simple search can be repeated as needed:

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

2. Next, stop the Directory Server instance on ds-east-01.example.com using the `stop-server` command and immediately retry the above searches. There should be no errors or noticeable delay in processing the search.

```
root@ds-east-01: bin/stop-server

root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. Restart the Directory Proxy Server instance on ds-east-01.example.com. Check the access log to confirm that the Directory Proxy Server started to include the ds-east-01 server in load-balancing within 30 seconds. The default time is 30 seconds, though you can change this default if desired.

Expanding the Deployment

In the following example deployment, the PingDirectory Server is deployed in a third, centrally-located data center. The directory servers in the central city is assigned to a new location named central. The proxies will use StartTLS to communicate with the directory servers in the central region.

Note: Other than the ability to add to the Directory Proxy Server's truststore, the `prepare-external-server` tool does not alter the Directory Proxy Server configuration in any way.

The Directory Proxy Server itself, installed on proxy-east-01.example.com, remains in the East location. This example will reconfigure load balancing between the six directory servers in three locations:

ds-east-01.example.com
ds-east-02.example.com
ds-west-01.example.com
ds-west-02.example.com
ds-central-01.example.com
ds-central-02.example.com

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Prepare the new external servers using the `prepare-external-server` tool.
- Use the `dsconfig` tool to configure the new LDAP external servers in the central data center and reconfigure the load-balancing algorithm to take these servers into account.
- Test external server communications after the servers have been configured and test a simulated external server failure.

Preparing Two New External Servers Using the prepare-external-server Tool

First, we prepare the external directory servers, ds-central-01 and ds-central-02, by creating the proxy user account and the supporting access rules. In this example, we will connect to the ds-central-01 PingDirectory Server using StartTLS. Because we are using StartTLS, we need to capture the ds-central-01 server's certificate and put it in the trust store on our Directory Proxy Server instance.

The `prepare-external-server` tool is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy. In this example, we run the tool on the ds-east-01 instance of the Directory Proxy Server.

To Prepare Two New External Servers Using the prepare-external-server Tool

1. Run the `prepare-external-server` tool to prepare the two new servers. On the first attempted bind to the server, the tool will report a "failed to bind" message as it cannot bind to the `cn=Proxy User` entry due to its not being created yet. The tool sets up the `cn=Proxy User` entry so that the Directory Proxy Server can access it and tests the communication settings to the server.

```
root@proxy-east-01: ./prepare-external-server \
--hostname ds-central-01.example.com --port 389 \
--baseDN dc=example,dc=com \
--proxyBindPassword password  \
--useStartTLS \
--proxyTrustStorePath ../config/ExampleTruststore.jks
```

```
Failed to bind as ‘cn=Proxy User’

Would you like to create or modify root user ‘cn=Proxy User” so that it is
available for this Directory Proxy Server? (yes / no)[yes]:

Enter the DN of an account on ds-central-01:389 with which to create or
 manage the ‘cn=Proxy User’
account [cn=Directory Manager]:

Enter the password for ‘cn=Directory Manager’:

Created ‘cn=Proxy User,cn=Root DNs,cn=config’
Testing ‘cn=Proxy User’ privileges ....Done
```

2. Repeat the process on the other new server in the central location, ds-central-02.

Note: For entry-balancing deployments, the global base DN is required when using `prepare-external-server`.

Adding the New PingDirectory Servers to the Directory Proxy Server

After preparing the external PingDirectory Servers to communicate with the Directory Proxy Server, we can now add the two servers in the central location to the proxy server instance. Because we have run the `prepare-external-server` tool, the two servers have the `cn=Proxy User` entry configured.

To Add the New PingDirectory Servers to the Directory Proxy Server

- Run the `dsconfig` tool, which is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy.

```
root@proxy-east-01:./dsconfig

>>>> Specify LDAP connection parameters

Directory Proxy Server hostname or IP address [localhost]:

How do you want to connect to the Directory Proxy Server at
localhost?

    1)  LDAP
    2)  LDAP with SSL
    3)  LDAP with StartTLS

Enter choice [1]: 1

Directory Proxy Server at localhost port number [389]:
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

Adding New Locations

First, we add a new central location, to which our new PingDirectory Servers will be added.

To Add a New Location

The following steps show how to add the new servers to a new location using `dsconfig` interactive.

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.

```
$ bin/dsconfig
```

2. On the main menu, enter the number corresponding to **Location**.
3. On the Location menu, enter the number corresponding to creating a new location.
4. Enter the option to create a new location from scratch.

5. Configure the `preferred-failover-location` property of the new location so that this location fails over first to the east location and then to the west location, should all of the servers in the central location become unavailable.
6. Add the east and west locations as values of the property, specifying them in the order that they will be used for failover.
7. Confirm that these are the correct values and finish configuring the location.

Editing the Existing Locations

Next, we edit the existing east and west locations to include the new central location in their failover logic. The new failover logic will be based on geographic distance, so that the east location will first fail over to central and then the west location.

To Edit Existing Locations

The following example procedure uses `dsconfig` interactive mode to edit the east location.

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. On the Directory Proxy Server console configuration menu, enter the number corresponding to Location.
3. On the Location menu, enter the number corresponding to viewing and editing an existing location. Then, enter the number corresponding to the Location to be changed.
4. Remove the west location from the `preferred-failover-location` property. It will be added later.
5. Add a new value to the `preferred-failover-location` property.
6. Select the values of the new failover locations for the east.
7. Confirm the new configuration information and save the changes.
8. Repeat steps 2-7 to reconfigure the failover logic for the west location to include the new central location.
9. List the locations to confirm that the new location was added correctly.

Adding New Health Checks for the Central Servers

Next, we must add new health checks for the two new servers.

To Add New Health Checks for the Central Servers

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. Select the number corresponding to creating a new health check.
3. Enter the option to use an existing health check as a template.
4. Enter the number corresponding to the ds-east-01 health check to use it as a template for the new health check.
5. Name the new health check using the same naming strategy established for the other servers in the deployment. As this health check is for the ds-central-01 server, the name takes the following format:

```
>>>> Enter a name for the Search LDAP Health Check that you want to create:
ds-central-01.example.com:389_dc_example_dc_com-search-health-check
```

6. Review the configuration properties and then enter `f` to finish configuring the new health check and save changes.
7. Repeat steps 2-6 to create another new health check for the `ds-central-02` server.

Adding New External Servers

Add new external servers by selecting “External Server” from the main menu.

To Add New External Servers

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. On the External Server menu, enter the number corresponding to "Create a new External Server".

3. Base the configuration of the new external server on the existing configuration of the ds-east-01 server. Enter `t` to use an existing External Server as a template.
4. Enter the number to base the configuration of the new server on the configuration of the ds-east-01 server.
5. Enter a name for the new ds-central-01 server that complies with the naming strategy.

   ```
   >>>> Enter a name for the Ping Identity DS External Server that you
   want to create: ds-central-01.example.com:389
   ```
6. Enter the value of the `server-host-name` property.
7. Review and modify the configuration properties of the external server.
8. On the External Server menu, change the `server-host-name` property to reflect the name of the `ds-central-01` server.
9. On the External Server menu, change the `location` property to reflect the central location.
10. Change the `health-check` property to reflect the new health check created for the ds-central-01 server in the previous section.
11. On the 'health-check' Property menu, enter the number to remove one or more values.
12. Add the health-check created in the previous section.
13. Select the health check associated with the ds-central-01 server.
14. Press **Enter** to use the value associated with ds-central-01 health check.
15. Review the configuration of the new external server and enter `f` to create the server.
16. Repeat these steps to add the new `ds-central-02` external server.

Modifying the Load Balancing Algorithm

To modify the existing load-balancing algorithm to include the newly created servers, select “Load-Balancing Algorithm” from the main menu.

To Modify the Load-Balancing Algorithm

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. Choose the option for Load-Balancing Algorithm.
3. On the Load-Balancing Algorithm menu, enter the number corresponding to "View and edit an existing Load-Balancing Algorithm".
4. Add the ds-central-01 and ds-central-02 servers to the `backend-server` configuration property.
5. On the backend-server property menu, enter the number corresponding to adding one or more values.
6. Select the external servers to add. In this example, select ds-central-01.example.com and ds-central-02.example.com.
7. Review the changes made to the load-balancing algorithm’s configuration properties, and enter `f` to save changes.

   The change has been saved and applied to the Directory Proxy Server. The load-balancing algorithm is referenced in the load-balancing-algorithm property of the request processor used by this Directory Proxy Server.
8. To view this property, go to the main menu and select the Request Processor option.
9. On the Request Processor menu, enter the number corresponding to view and edit an existing request processor.
10. Select the request process used by the Directory Proxy Server, and review the configuration properties.

    This request processor is used by the subtree view serviced by the Directory Proxy Server, which is in turn referenced by the client connection policy.

    Note: The changes made in this procedure are already in effect. The Directory Proxy Server does not have to be restarted.

Testing External Server Communication

After adding and configuring the new external servers, test the communication between the Directory Proxy Server and the LDAP external servers using the `include-backend-server-passthrough-subtree-views` property of the Directory Proxy Server in combination with an LDAP search. For more information about this option, see Testing External Server Communications on page 190.

To Test External Server Communication

- Run the `ldapseasrch` command to test communications on the ds-central-01 serverTask.

```
root@proxy-east-01: bin/ldapsearch --port 389 --bindDN "cn=directory
 manager" \
--bindPassword password \
--baseDN "dc=example,dc=com,ds-backend-server=ds-central-01.example.com:389"
 \
--searchScope base "(objectclass=*)"
```

You can repeat this search on the ds-central-02 server, to confirm that the server returns the entry as expected.

Testing a Simulated External Server Failure

Once you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the Directory Proxy Server redirects LDAP requests appropriately. We stop the ds-east-01.example.com:389 and ds-east-02.example.com:389 server instances and test searches through proxy-east-01.example.com.

To Test a Simulated External Server Failure

1. We stop the ds-east-01.example.com:389 and ds-east-02.example.com:389 server instances and test searches through proxy-east-01.example.com.
2. Perform several searches against the Directory Proxy Server. Verify activity in each of the servers in the east location, ds-east-01 and ds-east-02, by looking at the access logs. The following simple search can be repeated as needed:

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. Next, stop the Directory Server instance on ds-east-01.example.com and ds-east-02.example.com using the stop-server command and immediately retry the above searches. There should be no errors or noticeable delay in processing the search.

```
root@proxy-east-01: bin/stop-server

root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" --bindPassword password \
--baseDN "dc=example,dc=com" --searchScope base --useStartTLS \
"(objectclass=*)"
```

4. Check the access log to confirm that requests made to these servers are routed to the central servers, as these servers are the first failover location in the failover list for the ds-east-01 and ds-east-02 servers.
5. Restart the Directory Server instance on ds-east-01.example.com and ds-east-02.example.com. Check their access logs to ensure that traffic is redirected back from the failover servers.

Merging Two Data Sets Using Proxy Transformations

In the following example, the Example.com company acquires Sample Corporation. During the merger, Example.com migrates data from Sample's `o=sample` rooted directory, converting Sample's sampleAccount auxiliary object class usage to Example.com's exampleAccount object class for entries rooted under `dc=example,dc=com`. Knowing that it can take considerable time for Sample's directory clients to become aware of the new DIT and schema, proxy data transformations are created to give the Sample clients as consistent a view of the data as possible during the migratory period. These transformations allow the clients to search and modify entries under `o=sample` using the Sample Corp. schema.

Overview of the Attribute and DN Mapping

To achieve the merger of the two data sets, we create proxy transformations that map the Sample source attributes to Example.com target attributes as described in Table 9-1, “Attribute Mapping”. The Example.com schema already defines an attribute to contain the RDN of user entries, called `uid`. However, Example.com chooses to create two new attributes within its `exampleAccount` object class to accommodate two attributes in the Sample schema for representing the region and the DN of linked accounts.

During the merger, Example.com decides to re-parent Sample’s customer entries, which are defined under two different subtrees, ou=east,o=sample and ou=west,o=sample, placing them under Example.com’s `ou=people,dc=example,dc=com` subtree. Associated proxy transformations are described in Table 9-2, "DN Mappings". In this process, Example.com collapses the Sample tree, moving entries from the east and west region under a single DN, `dc=example,dc=com`. The DN proxy transformations assume that all Sample users have been co-located under this single Example.com subtree.

Table 9: Attribute Mapping

Sample Attribute	Example.com Attribute	Description
sampleID	uid	RDN of user entries
sampleRegion	exSampleRegion	String value representing the region
sampleLinkedAccounts	exSampleLinkedAccounts	DN value

Legacy Sample LDAP applications searching for entries in either the Sample base DN `ou=east,o=sample` or `ou=west,o=sample` will be successfully serviced, though there will be one or more differences in the user entries seen by the Sample legacy applications. Since the Example.com Directory Server has no knowledge of the Sample user’s former `ou=east` or `ou=west` association, search results for client searching under `o=sample` will return a DN that may differ from the original search base. For instance, a search for `sampleID=abc123` under `ou=west,o=sample` may return the user entry for abc123 with the DN of `sampleID=abc123,ou=east,o=sample`. The following table illustrates the mapping DNs.

Table 10: DN Mapping

Sample DN	Example.com DN
ou=east,o=sample	dc=example,dc=com
ou=west,o=sample	dc=example,dc=com
o=sample	dc=example,dc=com

About Mapping Multiple Source DNs to the Same Target DN

Some complications exist when defining multiple DN mappings that are used for the same request processor and the same source or target DN (or that have source or target DNs that are hierarchically related). The client request may not include enough information to disambiguate and determine the proper rule to follow.

Several solutions exist to avoid problems of disambiguation. If the client does not need to be able to see all mappings at the same time, then a new client connection policy can be created to use connection criteria that select the set of mappings applied to the client based on information such as the IP address or bind DN. Each client connection policy would have separated subtree views with separate proxying request processors that reference the appropriate transformation for that client.

Alternatively, if it is unnecessary to search under the o=sample base DN, then separate subtree views can be created in the same client connection policy. For example, one subtree view would be created for `ou=east,o=sample` and one for `ou=west,o=sample`. Each subtree view is then associated with its own proxying request processor, one for ou=east requests and one for ou=west requests.

An Example of a Migrated Sample Customer Entry

The following example is an example of a Sample customer entry that has been migrated to the Example.com database. The user entry is defined in the Example.com Directory Server's database as follows. The attributes that have undergone a proxy transformation are marked in bold. Note that this view is how the entry appears to search requests under the `dc=example,dc=com` base DN.

```
dn: uid=scase,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
exAccountNumber: 234098
exSampleRegion: east
exSampleLinkedAccounts: uid=jcase,ou=people,dc=example,dc=com
userPassword: password
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
l: Arlington
mail: sterlingcase@maildomain.com st: VA
```

The following examples shows what the Directory Proxy Server returns to LDAP clients who have requested the entry when searching under the o=sample base DN. Note that the DN returned includes ou=east, even though this branch does not exist in the Example.com DIT. It also returns the attribute names as they are defined in the Sample schema.

```
dn:  sampleID=scase,ou=east,o=sample
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
exAccountNumber: 234098
exSampleRegion: east
exSampleLinkedAccounts: sampleID=jcase,ou=people,dc=example,dc=com
userPassword: password
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
l: Arlington
mail: sterlingcase@maildomain.com st: VA
```

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Install any necessary schema on the Directory Proxy Server.
- Create three attribute mapping proxy transformations and three DN mapping proxy transformations
- Create a new proxying request processor, using the existing `dc_example_dc_com` request processor as a template.
- Assign the six proxy transformations to the new proxying request processor.

- Create a new subtree view for o=sample that references the new proxying request processor.
- Add the new subtree view to the existing client connection policy.
- Test our configuration by performing some searches on the Sample DIT.

About the Schema

The Directory Proxy Server inherits user-defined schema from all external servers by comparing `cn=schema` on these servers at Directory Proxy Server startup and at five minute intervals. As a result, example.com schema does not need to be added manually to the Directory Proxy Server's `config/schema` directory. We assume that the schema for Sample entries has been defined on the external servers with the example.com DIT, requiring no direct schema management on the Directory Proxy Server. The following schema definitions are assumed to exist on the external Directory Server:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.1
  NAME 'exAccountNumber'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.3
  NAME 'sampleLinkedAccounts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.2
  NAME 'sampleRegion'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.1
  NAME 'sampleID'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.3
  NAME 'exSampleLinkedAccounts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.2
  NAME 'exSampleRegion'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
objectClasses: ( 1.3.6.1.4.1.32473.2.2.1
  NAME 'exampleAccount'
  SUP top
  AUXILIARY
  MAY ( exAccountNumber $
        exSampleRegion $
        exSampleLinkedAccounts $
        sampleID $
        sampleRegion $
        sampleLinkedAccounts ) )
```

The schema file defines some Example.com schema, such as `exAccountNumber` and `exSampleRegion`, and some Sample schema, such as `sampleRegion` and `sampleID`.

Creating Proxy Transformations

We create three attribute mapping proxy transformations and three DN mapping proxy transformations. We run the `dsconfig` tool, which is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy.

To Create Proxy Transformations

1. In the main server root directory, PingDirectoryProxy, run the `start-server` command.

```
$ bin/start-server
```

2. Run `dsconfig` in interactive mode and enter the LDAP connection parameters.
3. On the Configuration main menu, enter the number corresponding to **Proxy Transformation**.

Creating the Attribute Mapping Proxy Transformations

Next, we create the attribute mapping proxy transformations using `dsconfig` interactive. We assume for this example that we are continuing from the previous `dsconfig` session. In the following example, this transformation maps `ou=east,o=sample` in the Sample schema `dc=example,dc=com` in the Example.com schema.

To Creating the Attribute Mapping Proxy Transformations

1. On the Proxy Transformation menu, enter the number corresponding to "Create a New Proxy Transformation".
2. Create a mapping from the sampleRegion attribute to the `exSampleRegion` attribute, enter the number corresponding to "Attribute Mapping Proxy Transformation".

```
>>>> Select the type of Proxy Transformation that you want to create:

    1)  Attribute Mapping Proxy Transformation
    2)  Default Value Proxy Transformation
    3)  DN Mapping Proxy Transformation
    4)  Groovy Scripted Proxy Transformation
    5)  Simple To External Bind Proxy Transformation
    6)  Suppress Attribute Proxy Transformation
    7)  Suppress Entry Proxy Transformation
    8)  Third Party Proxy Transformation
```

3. Enter a descriptive name for the new proxy transformation that illustrates the attribute mapping that it performs.
4. Press **Enter** to enable the proxy transformation.
5. Provide the name of the source attribute in the Sample schema to map to the Example.com schema, which is sampleRegion.
6. Review the configuration properties, and enter `f` to create the new attribute mapping proxy transformation.
7. Repeat the previous steps to create another attribute mapping proxy transformation. This time, map between the Sample Corporation’s sampleID attribute and the Example.com uid attribute.
8. Repeat the previous steps again to create a last attribute mapping proxy transformation, mapping between the Sample sampleLinkedAccounts attribute and the Example.com exSampleLinkedAccounts attribute.

Creating the DN Mapping Proxy Transformations

Now we create the DN mapping proxy transformations.

To Create the DN Mapping Proxy Transformations

1. On the Proxy Transformation menu, enter the number corresponding to Create a new Proxy Transformation.
2. Enter the option to create a new Proxy Transformation from scratch.
3. Enter the option for "DN Mapping Proxy Transformation."
4. Enter a name for the DN Mapping Proxy Transformation. This transformation maps `ou=east,o=sample` in the Sample schema `dc=example,dc=com` in the Example.com schema.
5. Select `TRUE` to enable the transformation by default.
6. Specify the source DN as it appears in client requests.

```
>>>> Configuring the 'source-dn' property

     Specifies the source DN that may appear in client
     requests which should be remapped to the target DN.
     Note that the source DN must not be equal to the target DN.
     Syntax: DN

Enter a value for the 'source-dn' property:
ou=east,o=sample
```

7. Specify the target DN, where requests for the source DN should be routed.

```
>>>> Configuring the 'target-dn' property

    Specifies the DN to which the source DN should be mapped.
    Note that the target DN must not be equal to the source
    DN.

    Syntax: DN

Enter a value for the 'target-dn' property: dc=example,dc=com
```

8. Review the configuration properties, and then enter f to create the new DN mapping proxy transformation.
9. using the previous steps, create a new DN mapping proxy transformation that maps `ou=west,o=sample` in the Sample schema to `dc=example,dc=com` in the Example.com schema, and name it `sample_west-to-example`.
10. Finally, create a DN mapping proxy transformation for the base DN of the Sample database.

Creating a Request Processor to Manage the Proxy Transformations

Next, we need to create a new proxying request processor that includes our new attribute and DN mapping proxy transformations. We will use the existing `dc_example_dc_com` request processor as a template.

To Create a Request Processor to Manage Proxy Transformations

1. On the Configuration main menu, enter the number corresponding to Request Processor.
2. On the Request Processor menu, enter the number corresponding to "Create a new Request Processor."
3. Choose the option to use the current request processor as a template.
4. Provide a name for the new proxying request processor, such as `o_sample-req-processor`.
5. Review the properties. The load-balancing algorithm is the same as for the previous request processor, though the transformation property must be changed. Enter the number corresponding to the `Transformation` property.
6. Enter the number corresponding to the proxy transformations that we created in the previous sections.
7. Select the attribute mapping proxy transformations first. Next, select the DN mapping proxy transformations. The order of the selection is important because we have related DNs. Begin with the DNs that are lower in the tree first, and finish with the base DN transformation.

```
Select the Proxy Transformations you wish to add:

    1)  sample-to-example       5)  sampleLinkedAccounts-to-
                                    exSampleLinkedAccounts
    2)  sample_east-to-example  6)  sampleRegion-to-
                                    exSampleRegion
    3)  sample_west-to-example  7)  Create a new Proxy
                                    Transformation
    4)  sampleID-to-uid         8)  Add all Proxy Transformations

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 4,5,6,2,3,1
```

8. Confirm that the proxy transformations are listed in the correct order and press **Enter** to accept and use the values.
9. Review the request processor properties, and enter `f` to save changes.

Creating Subtree Views

At this stage, we need to configure subtree views for the Directory Proxy Server.

To Create Subtree Views

1. On the Configuration main menu, enter the number corresponding to Subtree View.
2. On the Subtree View menu, enter the number corresponding to "Create a new Subtree View."
3. Enter the option to create the new subtree view from an existing one.
4. Select the `dc_example_dc_com-view` subtree view.
5. Enter a descriptive name for the subtree view configuration.
6. Configure the base DN property of the Sample dataset.
7. Enter the request processor created in the previous section.
8. Review the configuration properties, and enter `f` to save changes.

```
>>>> Configure the properties of the Subtree View

        Property           Value(s)
        ------------------------------------------------
    1)  description        -
    2)  base-dn            "o=sample"
    3)  request-processor  o_sample-req-processor

    ?)  help
    f)  finish - create the new Subtree View
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q) quit
```

Editing the Client Connection Policy

Finally, we edit the client connection policy to add our new `o=sample` subtree view.

To Edit the Client Connection Policy

1. On the Configuration main menu, enter the number corresponding to Client Connection Policy.
2. On the Client Connection menu, enter the number corresponding to "Create a new Client Connection."
3. In the configuration properties, select the `subtree-view` property. Enter the number corresponding to "Add one or more values" to add the new subtree view created for the previous example.
4. Select the subtree view that was created in the previous section.

```
Select the Subtree Views you wish to add:

    1)  o_sample-view
    2)  Create a new Subtree View
```

5. Review the subtree views now referenced by the property and press **Enter** to use these values.
6. Review the configuration properties of the client connection policy and enter `f` to save changes.

Testing Proxy Transformations

After setting up the deployment scenario, the Directory Proxy Server will now respond to requests to the `dc=example,dc=com` and o=sample base DNs. We now test the service by imitating example client requests to search and modify users.

Testing Proxy Transformations

The following example fetches the user with sampleID=scase under the `ou=east,o=sample` base DN.

1. Run `ldapsearch` to view a Sample entry.

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "ou=east,o=sample" "(sampleID=scase)"
   ```

   ```
   dn: sampleID=scase,ou=People,ou=east,o=sample
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   objectClass: exampleAccount
   objectClass: top
   description: A customer account migrated from Sample merger
   sampleID: scase
   userPassword: {SSHA}A5O4RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZXA==
   exAccountNumber: 234098
   givenName: Sterling
   cn: Sterling Case
   sn: Case
   telephoneNumber: +1 804 094 3356
   street: 00468 Second Street
   mail: sterlingcase@maildomain.com
   l: Arlington
   st: VA
   sampleRegion: east
   sampleLinkedAccounts: sampleID=jcase,ou=People,ou=east,o=sample
   ```
2. Modify the sampleRegion value, changing it to west. To do this, we first create a `ldapmodify` input file, called `scase-mod.ldif`, with the following contents:

   ```
   dn: sampleID=scase,ou=People,ou=east,o=sample
   changetype: modify
   replace: sampleRegion
   sampleRegion: west
   ```
3. Use the file as an argument in the `ldapmodify` command as follows.

   ```
   root@proxy-east-01: bin/ldapmodify --bindDN "cn=Directory Manager" \
   --bindPassword password --filename scase-mod.ldif
   ```

   ```
   Processing MODIFY request for sampleID=scase,ou=People, ou=east,o=sample
   MODIFY operation successful for DN sampleID=scase,ou=People,
    ou=east,o=sample
   ```
4. Search for scase's `sampleRegion` value under o=sample, we should see west:

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "o=sample"  "(sampleID=scase)" \
   sampleRegion
   ```

   ```
   dn: sampleID=scase,ou=People,ou=east,o=sample
   sampleRegion: west
   ```
5. Search for scase by `uid` rather than `sampleID`, under the `dc=example,dc=com` base DN. We see the Example.com schema version of the entry:

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "dc=example,dc=com"  "(uid=scase)"
   ```

   ```
   dn: uid=scase,ou=People,dc=example,dc=com
   objectClass: person
   objectClass: exampleAccount
   objectClass: inetOrgPerson
   objectClass: organizationalPerson
   objectClass: top
   ```

```
description: A customer account migrated from Sample merger
uid: scase
userPassword: {SSHA}A5O4RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZXA==
exAccountNumber: 234098
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
mail: sterlingcase@maildomain.com
l: Arlington
st: VA
exSampleRegion: west
exSampleLinkedAccounts: uid=jcase,ou=People,dc=example,dc=com
```

Chapter 6

Deploying an Entry-Balancing Directory Proxy Server

Topics:

You can deploy PingDirectoryProxy Server in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates an entry-balancing deployment scenario.

Deploying an Entry-Balancing Proxy Configuration

Entry-balancing is a Directory Proxy Server configuration that allows the entries within a portion of the Directory Information Tree (DIT) to reside on multiple external servers. This configuration is typically useful when the DIT contains many millions of entries, which can be difficult to bring completely into memory for optimal performance. Entry-balancing allows entries under a balancing point base DN to be divided among any number of separate directory servers, making the Directory Proxy Server responsible for intelligently routing requests based on the division.

In this example scenario, the entries in the DIT outside of the balancing point are replicated across all external servers known to the Directory Proxy Server. Replication on the external directory servers must be properly configured before proceeding through this example. The directory servers are expected to contain two replication domains: the global domain, `dc=example,dc=com`, and the balancing point, `ou=people,dc=example,dc=com`.

In this deployment scenario, an `austin-proxy1` instance of the Directory Proxy Server communicates with four external directory servers. The Directory Proxy Server is configured to use entry balancing for the `ou=people,dc=example,dc=com` base DN, with two sets of user entries split beneath it. The first set of user entries is defined in the replicated pair of external servers, `austin-set1.example.com` and `newyork-set1.example.com`. The second set of entries is defined in `austin-set2.example.com` and `newyork-set2.example.com`. The entries in the `dc=example,dc=com` DIT outside of the balancing point base DN are replicated among the four external servers.

The following `dsreplication status` output from the PingDirectory Server external servers describes the replication configuration that exists before creating the Directory Proxy Server configuration.

```
--- Replication Status for dc=example,dc=com: Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
--------:---------:--------:---------------------------:-------------

austin-set1.example.com:389  : 10003   : 0   : N/A  : 722087263
austin-set2.example.com:389  : 10003   : 0   : N/A  : 722087263
newyork-set1.example.com:389 : 10003   : 0   : N/A  : 722087263
newyork-set2.example.com:389 : 10003   : 0   : N/A  : 722087263

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset1):
 Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
--------:---------:---------:---------------------------:-----------------

austin-set1.example.com:389  : 100001  : 0  : N/A               : 178892712
newyork-set1.example.com:389 : 100001  : 0  : N/A               : 178892712

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset2):
 Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
---------:---------:---------:---------------------------:-----------------

austin-set2.example.com:389  : 100001  : 0       : N/A        : 1057593890
newyork-set2.example.com:389 : 100001  : 0       : N/A        : 1057593890
```

Determining How to Balance Your Data

If a single Directory Server instance can hold all of your data, then we recommend storing your data on a single server and replicating for high availability, as this simplifies your deployment. If a single server cannot hold all of your data, then you can spread it across multiple servers in several ways:

- If the data is already broken up by hierarchy and all of the clients understand how to access it that way, the number of top-level branches is small and a single Directory Server instance can hold all of the information within one or more branches. Configure the Directory Proxy Server with multiple base DNs and use simple load-balancing rather than entry balancing to simplify your deployment.

- If simply breaking up the data using the existing hierarchy is not an option, for example if a large number of top-level branches must be configured, then consider using entry balancing. The contents of any single branch still must fit on a given server, because only entries that are immediate subordinates of the entry-balancing base DN may be spread across multiple servers. Any entries that are further subordinates have to be placed in the same directory server instance as their parent.
- If one or more branches are so large that any single Directory Server instance cannot hold all of the data, you need to use entry balancing within that branch to divide the entries among two or more sets of Directory Servers. You may also need to change the way that the data is arranged in the server so that it uses as flat a DIT as possible, which is easier to use in an entry-balancing deployment.

In an entry-balancing deployment, there can be data that is common to all external directory servers outside the balancing point. This data is referred to as the global domain. The Directory Proxy Server entry-balancing configuration will contain at least two subtree views and associated request processors, one for the global domain and one for the entry-balancing domain. In our examples, the global domain is `dc=example,dc=com` and the entry-balancing domain is `ou=people, dc=example,dc=com`. The entry-balancing base DN, `ou=people,dc=example,dc=com`, is also the balancing point.

Entry Balancing and ACIs

In an entry-balancing deployment, access control instructions (ACIs) are still configured in the backend Directory Server data. When defining access controls in an entry-balancing deployment, you need to ensure that the data used by the access control rule is available for evaluation on all datasets.

If you use groups for access control and a group contains users from different data sets, then that group must exist on each dataset. For a single ACI to be applicable to entries in all datasets, it must be specified above the entry-balancing point. For example, if an ACI allows access to modify users that are part of group 1, then two things must exist on both data sets:

- Group 1 must exist in the `ou=groups` branch of both datasets.
- The ACI referencing group 1 must exist in the `ou=people` branch or above. The `ou=people` branch entry itself is part of the common data.

The Directory Proxy Server ensures that any changes to entries within the scope of the entry-balancing request processor, but outside the balancing point, are applied to all backend server sets. Any ACI stored at the entry-balancing point will be kept in sync if changes are made through the Directory Proxy Server.

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Install the Directory Proxy Server on `austin-proxy1`.
- Use the `create-initial-proxy-config` tool to provide our initial setup for entry balancing. The initial setup includes defining multiple subtree views and global indexes in support of entry balancing.
- Change the placement algorithm of the `austin-proxy-01` server to use an entry-count placement algorithm. This algorithm is used to select the backend set to which to forward an add request. It looks at the number of entries in the backend sets and forwards the add request to the backend with either the fewest or the most entries, depending on the configuration. You can also configure the placement algorithm to make the decision based on the on-disk database size rather than the number of entries.

Installing the Directory Proxy Server

We start by configuring the Directory Proxy Server. The four external servers, `austin-set1.example.com`, `newyork-set1.example.com`, `austin-set2.example.com`, and `newyork-set2.example.com`, are running.

To Install the Directory Proxy Server

- Run the `setup` program in non-interactive mode.

```
root@austin-proxy1: ./setup --acceptLicense \
--listenAddress austin-proxy1.example.com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --entryBalancing \
--aggressiveJVMTuning --maxHeapSize 2g --no-prompt
```

Configuring the Entry-Balancing Directory Proxy Server

Once the Directory Proxy Server has been installed, it can be automatically configured using the `create-initial-proxy-config` tool. This tool can only be used once for this initial configuration, after which we will have to use `dsconfig` to make any changes to our Directory Proxy Server configuration.

To Configure the Entry-Balancing Directory Proxy Server

1. Run the `create-initial-proxy-config` tool.

```
root@austin-proxy1: ./bin/create-initial-proxy-config
```

2. Our topology meets the requirements, press **Enter** to continue:

```
Some assumptions are made about the topology to keep
this tool simple:

1) all servers will be accessible via a single user account
2) all servers support the same communication security type
3) all servers are PingDirectoryProxy Servers

If your topology does not have these characteristics you can
use this tool to define a basic configuration and then use the
'dsconfig' tool or the Administrative Console to fine tune the
 configuration.

Would you like to continue? (yes / no) [yes]:
```

3. Provide the external server access credentials. All of our proxies have identical proxy user accounts and passwords.

```
Enter the DN of the proxy user account [cn=Proxy User,cn=Root
 DNs,cn=config]:

Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

4. Specify the type of security that the Directory Proxy Server will use to communicate with Directory Servers.
5. Enter a base DN of the Directory Server instances that will be accessed by the Directory Proxy Server.
6. Define the balancing point as a separate base DN, which is entry balanced:

```
Enter another base DN of the directory server instances that
will be accessed through the Directory Proxy Server:

    1)Remove dc=example,dc=com

    b)back

    q)quit

Enter a DN or choose a menu item [Press ENTER when finished
entering base DNs]: ou=people,dc=example,dc=com

Are entries within 'ou=people,dc=example,dc=com' split across
```

```
multiple servers so that each server stores only a subset of
the entries (i.e. is this base DN 'entry balanced')? (yes / no)
[no]: yes
```

7. In this example, the data in `ou=people,dc=example,dc=com` will be split across two backend sets. Enter 2 to specify that the data will be balanced across two sets of servers.

```
Across how many sets of servers is the data balanced?

    c) cancel creating ou=people,dc=example,dc=com
    q) quit

Enter a number greater than one or choose a menu item: 2
```

8. The balancing point is the same as our base DN, ou=people,dc=example,dc=com., so we use it as the entry balancing base.

```
>>>> Entry Balancing Base

The entry balancing base DN specifies the entry below which the
data is balanced. Entries not below this entry must be duplicated
in all the server sets. If all the entries in the base DN are
distributed the entry balancing base DN is the same as the base DN.

    c) cancel creating ou=people,dc=example,dc=com
    b) back
    q) quit

Enter the entry balancing base DN or choose a menu item
[ou=people,dc=example,dc=com]: ou=people,dc=example,dc=com
```

9. To improve the performance for equality search filters referencing the `uid` attribute, create a `uid` global index. Enter `yes` to add a new attribute to the global index.
10. Specify the `uid` attribute.

```
Enter attributes that you would like to add to the global index:

    c)cancel creating ou=people,dc=example,dc=com
    b)back
    q)quit

Enter an attribute name or choose a menu item [Press ENTER when
finished entering index attributes]: uid
```

11. To optimize Directory Proxy Server performance from the moment it starts accepting connections, enter the number corresponding to "Yes, and all subsequent attributes."
12. Press **Enter** to finish specifying index attributes.
13. Press **Enter** to enable RDN index priming.

```
Would you like to enable RDN index priming for
'ou=people,dc=example,dc=com'? (yes / no) [yes]:
```

14. Press **Enter** to finish specifying base DNs.

```
Enter another base DN of the directory server instances that
will be accessed through the Directory Proxy Server:

    1) Remove dc=example,dc=com
    2) Remove ou=people,dc=example,dc=com (distributed)

    b)  back
    q)  quit

Enter a DN or choose a menu item [Press ENTER when finished
```

```
entering base DNs]:
```

15. The external servers are spread among two locations, New York and Austin. This Directory Proxy Server instance is located in the austin location.

```
A good rule of thumb when naming locations is to use the
name of your data centers or the cities containing them.

    b)  back
    q)  quit

Enter a location name or choose a menu item: austin

    1)  Remove austin

    b)  back
    q)  quit
```

16. Define the newyork location:

```
Enter another location name or choose a menu item [Press ENTER
when finished entering locations]: newyork

    1)  Remove austin
    2)  Remove newyork

    b)  back
    q)  quit

Enter another location name or choose a menu item [Press ENTER
when finished entering locations]:
```

17. Select the austin location for this Directory Proxy Server instance:

```
Choose the location for this Directory Proxy Server

    1) austin
    2) newyork

    b) back
    q) quit

Enter choice [1]:
```

18. Specify the LDAP external server instances associated with this location.

```
Enter the host and port (host:port) of the first directory server
in 'austin'

     b)  back
     q)  quit

Enter a host:port or choose a menu item [localhost:389]:
austin-set1.example.com:389
```

19. Specify that the `austin-set1` server can handle requests from the global domain and from set 1 restricted domain.

```
Assign server austin-set1.example.com:389 to handle requests for
one or more of the defined sets of data:

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,2
```

20. Enter the number corresponding to "Yes, and all subsequent servers" to prepare the server for access by the Directory Proxy Server.

```
Would you like to prepare austin-set1.example.com:389 for access
by the Directory Proxy Server?

       1)Yes
       2)No
       3)Yes, and all subsequent servers
       4)No, and all subsequent servers

Enter choice [3]:
```

21. Select the entry-balanced data set that the austin-set1 server replicates with other servers.

```
You may choose a single entry-balanced data set with which
austin-set1.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 1
    2) None, data will not be replicated

Enter choice: 1

Testing connection to austin-set1.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ....Denied
```

22. Modify the root user for use by the Directory Proxy Server, specifying the directory manager password for the initial creation of the proxy user.

```
Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Enter the DN of an account on austin-set1.example.com:389
with which to create or manage the 'cn=Proxy User,cn=Root DNs,
cn=config' account and configuration [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':
Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config'privileges...Done
Setting replication set name .....
```

23. Since the replication set name has already been configured, we do not need to use the name created automatically by the Directory Proxy Server.

```
This server is currently configured for replication set 'dataset1'.
Would you like to reconfigure this server for replication set
'set-1'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done
Testing 'cn=Proxy User' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

24. Define the other Austin and New York servers using the same procedure as in the previous example:

```
Enter another server in 'austin'

    1) Remove austin-set1.example.com:389
    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]: austin-set2.example.com:389
```

```
Assign server austin-set2.example.com:389 to handle requests
for one or more of the defined sets of data

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,3

You may choose a single entry-balanced data set with which
austin-set2.example.com:389 will replicate data with other
servers

    1) ou=people,dc=example,dc=com; Server Set 2
    2) None, data will not be replicated

Enter choice: 1

Testing connection to austin-set2.example.com:389 ....Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Would you like to use the previously entered manager credentials
to access all prepared servers? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset2'.

Would you like to reconfigure this server for replication set 'set-2'?
(yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'austin'

    1) Remove austin-set1.example.com:389
    2) Remove austin-set2.example.com:389

    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:

>>>> >>>> Location 'newyork' Details
 >>>> External Servers

External Servers identify directory server instances including
host, port, and authentication information.

Enter the host and port (host:port) of the first directory server
in 'newyork':

    b) back
    q) quit
```

```
Enter a host:port or choose a menu item [localhost:389]:
newyork-set1.example.com:389

Assign server newyork-set1.example.com:389 to handle requests
for one or more of the defined sets of data

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,2

You may choose a single entry-balanced data set with which
newyork-set1.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 1
    2) None, data will not be replicated

Enter choice: 1

Testing connection to newyork-set1.example.com:389 ....Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset1'.

Would you like to reconfigure this server for replication set
'set-1'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'newyork'

    1) Remove newyork-set1.example.com:389
    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]: newyork-set2.example.com:389

Assign server newyork-set2.example.com:389 to handle requests
for one or more of the defined sets of data:

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,3

You may choose a single entry-balanced data set with which
new-york-set2.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 2
    2) None, data will not be replicated
```

```
Enter choice: 1

Testing connection to newyork-set2.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access.... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this Directory
Proxy Server? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config' Testing
'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset2'.
Would you like to reconfigure this server for replication
set 'set-2'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'newyork'

    1)Remove newyork-set1.example.com:389
    2)Remove newyork-set2.example.com:389

    b)back
    q)quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:

>>>> >>>> Configuration Summary

  External Server Security: None
  Proxy User DN: cn=Proxy User,cn=Root DNs,cn=config
  Location austin
    Failover Order: newyork
    Servers: austin-set1.example.com:389,
             austin-set2.example.com:389
  Location newyork
    Failover Order: austin
    Servers: newyork-set1.example.com:389,
             newyork-set2.example.com:389
  Base DN: dc=example,dc=com
    Servers: austin-set1.example.com:389,
             austin-set2.example.com:389,
             newyork-set1.example.com:389,
             newyork-set2.example.com:389
  Base DN:vou=people,dc=example,dc=com
    Entry Balancing Base: ou=people,dc=example,dc=com
    Server Set 1: austin-set1.example.com:389,
                  newyork-set1.example.com:389
    Server Set 2: austin-set2.example.com:389,
                  newyork-set2.example.com:389
    Index Attributes: uid (primed,unique)
    Prime RDN Index: Yes

    NOTE: The Directory Proxy Server must be restarted after
    this tool has completed to have index priming take place

       b) back
       q) quit
       w) write configuration

    Enter choice [w]
    >>>> Write Configuration

    The configuration will be written to a 'dsconfig' batch
    file that can be used to configure other Directory Proxy Servers.

    Writing Directory Proxy Server configuration to /proxy/dps-
cfg.txt.....Done
```

25. Enter `yes` to apply our configuration changes to the Directory Proxy Server.

```
Apply these configuration changes to the local Directory Proxy
Server? (yes /no) [yes]:

How do you want to connect to the Directory Proxy Server on localhost?

    1) LDAP
    2) LDAP with SSL
    3) LDAP with StartTLS

Enter choice [1]:

Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
Creating Locations ..... Done
Updating Failover Locations ..... Done
Updating Global Configuration ..... Done
Creating Health Checks ..... Done
Creating External Servers ..... Done
Creating Load-Balancing Algorithm for dc=example,dc=com .... Done
Creating Request Processor for dc=example,dc=com ..... Done
Creating Subtree View for dc=example,dc=com ..... Done
Updating Client Connection Policy for dc=example,dc=com ..... Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server
 Set 1 ..... Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set
 1...Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server
 Set 2 .... Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set
 2...Done
Creating Entry Balancing Request Processor for
 ou=people,dc=example,dc=com ..... Done
Creating Placement Algorithm for ou=people,dc=example,dc=com .... Done
Creating Global Attribute Indexes for ou=people,dc=example,dc=com ..... Done
Creating Subtree View for ou=people,dc=example,dc=com ..... Done
Updating Client Connection Policy for ou=people,dc=example,dc=com ..... Done

See /logs/create-initial-proxy-config.log for a detailed log of this
 operation

To see basic server configuration status and configuration you can launch /
bin/status
```

Configuring the Placement Algorithm Using a Batch File

Now, we configure the placement algorithm using a batch file. We want to place new entries added through the proxy via LDAP ADD operations into the least used dataset. We do this using an entry-count placement algorithm. To change the placement algorithm from round-robin to entry-count, we first create and enable an entry-count placement algorithm configuration object and then disable the existing round-robin placement algorithm. Our batch file, `dsconfig.post-setup`, contains the `dsconfig` commands required to create the entry-count placement algorithm and disable the old round-robin algorithm.

To Configure the Placement Algorithm Using a Batch File

The batch file contains comments to explain each `dsconfig` command. Note that in this example, line wrapping is used for clarity. The `dsconfig` command requires that the full command be provided on a single line.

The batch file itself looks like the following:

```
root@austin-proxy1:more ../dsconfig.post-setup

# This dsconfig operation creates the entry-count placement
# algorithm with the default behavior of adding entries to the
# smallest backend dataset first.

dsconfig create-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name entry-count --type entry-counter --set enabled:true

# Note that once the entry-count placement algorithm is created
# and enabled, we can disable the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then disable the
# original round-robin algorithm created during the setup
# procedure.

dsconfig set-placement-algorithm-prop
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin --set enabled:false

# At this point, LDAP ADD operations will be forwarded to an external
# server representing the dataset with the least number of entries.
```

- Run the `dsconfig` command using the batch file. Once the batch file has executed, a new entry-count placement algorithm, called entry-count, has been created, and the old round-robin placement algorithm, `round-robin`, has been disabled.

  ```
  root@austin-proxy1: bin/dsconfig --no-prompt \
  --bindDN "cn=directory manager" --bindPassword password \
  --port 389  --batch-file ../dsconfig.post-setup
  ```

  ```
  Batch file '../dsconfig.post-setup' contains 2 commands

  Executing: create-placement-algorithm --no-prompt
  --bindDN "cn=directory manager" --bindPassword pass
  --port 1389
  --processor-name ou_people_dc_example_dc_com-eb-req-processor
  --algorithm-name entry-count --type entry-counter --set enabled:true

  Executing: delete-placement-algorithm --no-prompt
  --bindDN "cn=directory manager" --bindPassword pass
  --port 1389
  --processor-name ou_people_dc_example_dc_com-eb-req-processor
  --algorithm-name round-robin --set enabled:false
  ```

Rebalancing Your Entries

If your deployment distributes entries using an entry counter placement algorithm or 3rd party algorithm, you may need to redistribute your entries. For example, imagine that you have an environment that distributes entries across three backends using an entry counter placement algorithm. This algorithm distributes entries to the backend that has the most space. Imagine that the backends all reach their maximum capacity and you decide to add a new backend to the deployment. You need to move the entries from the full backends and distribute them evenly across all the backends, including the new backend.

You might also want to deliberately rebalance your entries to meet the needs of your organization. For example, you can direct entry balancing based on attributes on the entries themselves. You can write a custom algorithm that looks at the value of an attribute that is being modified on the entry. Based on the attribute, you can then put this entry somewhere specific. You might use this feature if you want to have certain entries closer geographically to the client application using them. The geographical information could be included in the entry. Rebalancing would be used to move these entries to the server in the correct geographical location.

You can redistribute entry-balanced entries in two ways:

- **Using dynamic rebalancing.** With dynamic rebalancing, as existing entries get modified, they get moved. You configure dynamic rebalancing in the entry counter placement algorithm.
- **Using the `move-subtree` tool.** This tool can be used to move either small subtrees through a transactional method or to move large subtrees, potentially taking them offline for a short period.

The remainder of this section describes each of these method of entry rebalancing in more detail.

About Dynamic Rebalancing

During dynamic rebalancing entries get moved as they are modified. You configure dynamic rebalancing on the entry counter placement algorithm or a third-party placement algorithm that supports rebalancing. This algorithm keeps a count of the number of entries or the size of the backend set. You configure dynamic rebalancing using the following parameters:

- **rebalancing-enabled**. Determines whether entry rebalancing is enabled. When rebalancing is enabled, the placement algorithm is consulted after modify and add operations, to determine whether the target entry should be moved to a different backend set.
- **rebalancing-scope**. Indicates which modified entries are candidates for rebalancing. A value of top-level indicates that only entries immediately below the entry-balancing base can be rebalanced. A value of any indicates that entries at any level below the entry-balancing base may be rebalanced.
- **rebalancing-minimum-percentage**. Specifies the minimum threshold for entries to be migrated from one backend set to a preferred backend set with a smaller size. Entries are not migrated unless the percentage difference between the value of the current backend set and the value of the preferred backend set exceeds this threshold. This parameter prevents unnecessarily migrating entries back and forth between backend sets of similar sizes.
- **rebalancing-subtree-size-limit**. Specifies the maximum size of a subtree that can be rebalanced.
- **poll-interval**. Specifies how long to wait between polling the size of the backends to determine how to rebalance; works in conjunction with the `rebalancing-minimum-percentage` property.
- **placement-criteria**. Determines which approach to use to select a destination backend for rebalancing. Possible values are: `entry-count`, `backend-size`, or `custom`.

The following figure illustrates an entry-balancing base DN and three subtrees, A, B, and C. If the rebalancing scope is set to `any`, any child entries under the base DN can be rebalanced. For example, if a change is made to entry A1, the entire subtree A might be rebalanced, depending upon how you have configured rebalancing. If the rebalancing scope is set to `top-level`, rebalancing is only triggered when entries at the top level, such as A, are modified. Changes made to subentries, such as A1 or A2, do not trigger rebalancing. Rebalancing is also triggered upon the addition of entries such as `A1,A2`, provided the scope is `any`.

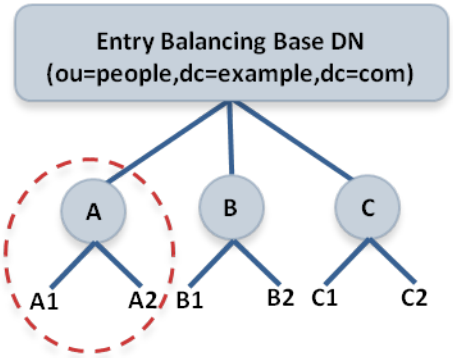


Figure 4: Rebalancing at the Top Level

If you are writing your own 3rd party algorithm, you program dynamic rebalancing using the SelectRebalancingBackendSet method on the placement algorithm. For more information, see the Server SDK documentation.

To Configure Dynamic Rebalancing

This procedure describes how to configure dynamic rebalancing on an existing entry balancing configuration.

1. To configure entry rebalancing, you may create an entry counter placement algorithm, if the current placement algorithm does not support rebalancing. You can either do this using `dsconfig` in interactive mode, or using the `dsconfig` command line as follows:

```
$ dsconfig create-placement-algorithm \
  --processor-name dc_example_dc_com-eb-req-processor \
  --algorithm-name rebalancing --type entry-counter \
  --set enabled:true --set rebalancing-enabled:true
```

2. Remove any placement algorithm previously configured on this entry-balancing request processor.
3. You can throttle how many entries are being moved by the proxy, so that the backend servers do not have too heavy a load. To do this, set the rebalancing-queue-maximum-size property of the request processor created in the previous step. By default, it is set to 1000. If the load is too high, reduce this value as follows:

```
$ dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set rebalancing-queue-maximum-size:50
```

4. Verify that the access logs are configured to display the subtrees being moved by dynamic rebalancing. The access logs provide a good way to monitor progress. So, if the write load on the backend servers is high and you see lots of rebalancing activity in the access log, lower the queue size to lower the rebalancing activity. You can configure the access log to display entry rebalancing processing information as follows:

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set log-entry-rebalancing-requests:true
```

About the move-subtree Tool

The `move-subtree` tool allows you to specify subtrees for rebalancing. You specify the source server, the target server, and one or more base DNs identifying the subtrees you want to move. You can move small subtrees using the transactional method or move large subtrees, which does not use this method. Instead, the large subtree is not fully accessible during the move, so clients may get an "insufficient access rights error" if they try to access the subtree. As entries are moved, clients can read but not write to them. Once the transfer is complete, the entries are fully available to client requests.

This tool accepts a file containing a list of the base DNs of the subtrees you want to move.

Note: The `move-subtree` tool requires users to have access to the extended operations and controls needed to run the tool. Make sure to apply the following ACIs to your data.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.5 ||
      1.3.6.1.4.1.30221.2.5.24 || 1.3.6.1.4.1.30221.2.5.13")
     (version 3.0; acl "Allow admin to submit move-subtree controls";
      allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (extop="1.3.6.1.4.1.30221.2.6.19")
     (version 3.0; acl "Allow admin to request move-subtree extended
      operation"; allow (read) userdn="ldap:///
uid=admin,dc=example,dc=com";)
```

About the subtree-accessibility Tool

The `subtree-accessibility` tool helps you determine if a subtree has restricted access and helps you fix any problems. If, during rebalancing, the Directory Server issues an alert that a subtree has been unavailable for too long, then you can use this tool to evaluate the problem. For example, if the `move-subtree` tool is interrupted by a host machine going down unexpectedly, the subtree might not be successfully moved. You can use the `subtree-accessibility` tool to evaluate and correct any problems with the subtrees, and then re-run the `move-subtree` tool.

Managing the Global Indexes in Entry-Balancing Configurations

In an entry-balancing configuration, the Directory Proxy Server maintains the default RDN index as well as one or more in-memory global attribute indexes. The global indexes allow the Directory Proxy Server to select the correct backend server set for incoming operations, which avoids broadcasting operations to all backend sets.

The indexes may be preloaded from peer proxies or the backend directory servers when the server starts up, and are updated by certain operations that come through the Directory Proxy Server. For instance, when a new entry is added, the DN of the new entry is added to the DN index of the Directory Proxy Server performing the operation. The indexes are also fault-tolerant and can adapt to changes made in the backend servers without going through the Directory Proxy Server. For example, operations will be processed directly through the backend server.

This section describes when to create a global attribute index, how to reload the global index, how to monitor its growth, and how to prime the global index from a peer at start-up.

When to Create a Global Attribute Index

The RDN index is referenced whenever a modify, delete, or base search is requested. In other words, the RDN index is needed when the LDAP request contains the complete DN of the targeted entry. If the entry-balancing request processor is not configured to prime the rdn index at startup, then the index is populated over time as LDAP requests are processed.

A global attribute index is an optional index and is referenced when the Directory Proxy Server is handling a search request with an equality filter involving the attribute, such as the `telephoneNumber` attribute with the filter (telephoneNumber=+11234567890). Since the Directory Proxy Server does not know what the data within the subtree views looks like or how it will be searched, it cannot create or recommend default global attribute index definitions. The creation of a global attribute index is based on the range of equality-filtered search requests that the Directory Proxy Server will handle. The Directory Server must also have an equality or ordering index type for the associated attribute Local DB Index."

The common candidates for global attribute indexing are the uniquely-valued equality-indexed attributes on the external servers. Examples of these attributes are `uid`, `mail` and `telephoneNumber`. Though the values of the attribute need not be unique to be used as a global attribute index by the entry-balancing request processor.

Consider a Directory Proxy Server deployment that expects to handle frequent searches of the form "`(&(mail=user@example.com)(objectclass=person))`". Since the filter is constructed with an equality match and &-clause, we can use a global attribute index on the `mail` attribute to avoid forwarding the search request to each entry balanced dataset.

The following `dsconfig` command creates the global attribute index. Note that the mail attribute must be indexed for equality searches on each of the external servers behind the Directory Proxy Server.

```
$ bin/dsconfig create-global-attribute-index \
  --processor-name ou_people_dc_example_dc_com-eb-req-processor \
  --index-name mail --set prime-index:true \
```

After creating the index with `dsconfig`, the index will begin to be populated as search requests involving the mail attribute are made to the Directory Proxy Server. At this point, you can also use the `reload-index` tool to fully populate the index for optimal performance as described in the following section.

Reloading the Global Indexes

The Directory Proxy Server provides a tool, `reload-index`, which allows you to manually reload the Directory Proxy Server global indexes. You might need to reload the index when:

- The Directory Proxy Server fails to successfully load its global indexes on startup.
- Changes are made to the set of indexed attributes.
- Significant changes are made to the content in backend servers.
- The integrity of the index is in question.

You can use the tool to reload all configured indexes in the global index, including the RDN index and all attribute indexes, or to reload only those indexes you specify.

The tool schedules an operation to run within the Directory Proxy Server's process. You must supply LDAP connection information so that the tool can communicate with the server through its task interface. Tasks can be scheduled to run immediately or at a later scheduled time. Once scheduled, you can manage the tasks using the `manage-tasks` tool.

To Reload All of the Index

- Run the `reload-index` tool to reload all of the indexes within the scope of the `dc=example,dc=com` base DN. The task is performed as `cn=Directory Manager` on port 389 of the localhost server. The existing index contents are erased before reloading.

```
$ bin/reload-index --task --bindPassword password --baseDN
 "dc=example,dc=com"
```

To Reload the RDN and UID Index

- To reload the RDN and UID index in the background so that the existing contents of these indexes can continue to be used, run the command as follows:

```
$ bin/reload-index --task --bindPassword password \
  --baseDN "dc=example,dc=com" --index rdn --index uid --background
```

To Prime the Backend Server Using the --fromDS Option

You can force the Directory Proxy Server to prime from the backend directory server only using the --fromDS option, overriding the configuration of the `prime-index-source` property. You can do this on a one off basis if the global index appears to be growing too large. For example, run the command as follows:

- Run the `reload-index` command with the --fromDS option to prime the backend server.

```
$ bin/reload-index --bindPassword password --baseDN "dc=example,dc=com" --
fromDS
```

Monitoring the Size of the Global Indexes

Over time, stale entries can build up in the global indexes because proxies do not communicate changes to the indexes with one another. The Directory Proxy Server continues to operate normally in this situation since the global indexes are only ever used as a hint at where to find entries.

The rate of this growth is typically very slow since in most environments the key attributes change infrequently. The global indexes themselves are also very compact. However, if the global indexes start to fill up the allocated memory, you may need to flush and reload them. The size of the global indexes can be monitored over LDAP using the following command:

```
$ bin/ldapsearch -b "cn=monitor" -D "uid=admin,dc=example,dc=com" -w password
 \
  "(objectClass=ds-entry-balancing-request-processor-monitor-entry)" \
  global-index-current-memory-percent
```

If the global indexes fill up, the Directory Proxy Server will continue to operate normally, but it will need to start evicting entries from the indexes, which will lead to more broadcast searches, reducing the overall throughput of the Directory Proxy Server.

To reload the indexes so that they no longer hold stale information, run the `reload-index` command with the --fromDS option so that data is loaded from backend directory servers. We recommend that you reload the indexes during off-peak hours because it may have an impact on performance while the reload is in progress.

Sizing the Global Indexes

The Directory Proxy Server includes a tool, `global-index-size`, to help you estimate the size in memory of your global indexes. You can estimate the size of more than one index in a single invocation by providing multiple sets of options. The tool creates its estimate using the following information:

- **Number of keys in the index**. For example, for the built-in RDN index, the number of keys is the total number of entries in the Directory Server that are immediately below the balancing point. Entries more than one level below the balancing point, as well as entries that are not subordinate to the balancing point, will not be contained in the RDN index. For attribute indexes, the number of keys will be the number of unique values for that attribute in the entry-balanced portion of the data.
- **Average size of each key, in bytes**. For attributes indexes, the key is simply the attribute value. For the built-in RDN index, the key is the RDN directly below the balancing base DN. For example, for the DN `uid=user.0,dc=example,dc=com` under the balancing base DN of `dc=example,dc=com`, the key size is 10 bytes (the number of bytes in the RDN uid=user.0).
- **Estimated number of keys**. This value corresponds to the maximum number of keys you expect in your Directory Server. The number of keys is provided in the index-size configuration property of the `global-attribute-index` object when you configure an attribute index. For the built-in RDN index, the configured number of keys is provided in the `rdn-index-size` property. If you do not provide a value, the tool assumes that the configured number of keys is the same as the actual number of keys.

To Size the Global Index

- Run the `global-index-size` to estimate the size of two separate indexes, both with 10,000,000 keys but with differing average key sizes. The configured number of keys is assumed to be equal to the actual number of keys:

```
$ bin/global-index-size --numKeys 10000000 \
  --averageKeySize 11 --numKeys 10000000 \
  --averageKeySize 15

Num Keys : Cfg. Num Keys : Avg. Key Size : Est. Memory Size
---------:---------------:---------------:-----------------
10000000 : 10000000      : 11            : 159 mb
10000000 : 10000000      : 15            : 197 mb
```

Priming the Global Indexes on Start Up

The Directory Proxy Server can prime the global indexes on startup from the backend directory server or from a peer proxy server, preferably one that resides on the same LAN or subnet. When priming occurs locally, you can avoid WAN bandwidth consumption and reduce the processing load on the directory servers in the topology. You can specify the data sources for the index priming and the order in which priming from these sources occurs.

Use the `prime-index-source` property to specify the sources of data, either `ds`, `file` or some combination of the two. The order you specify is the order in which priming from these sources will be attempted. For example, if you specify `prime-index-source:file,ds`, priming will be performed from the `global-index` data file created from the previous run of the directory servers. With the `file,ds` configuration, the contents of the global index are written to disk periodically if, and only if, the entire global index has been primed previously from a directory servers source either from startup or reloaded-index. Priming is most efficient if the source server is on the same local network as the Directory Proxy Server.

To Configure All Indexes at Startup

The following example configures the entry-balancing request processor so that it primes the global index from the persisted file, if present, or from an external directory servers source if necessary.

- Run the `dsconfig` tool to prime all indexes at startup.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set prime-all-indexes:true --set prime-index-source:file \
  --set prime-index-source:ds
```

To Configure the Global Indexes Manually

If you do not want to configure priming during setup, you can configure index priming manually by creating an external server, creating a global attribute index, and then changing the entry-balancing request processor to load indexes from this external server.

1. Use the `dsconfig` tool to create an external server of the type PingDirectoryProxy Server to represent a peer of the Directory Proxy Server.

```
$ bin/dsconfig create-external-server \
  --server-name intra-proxy-host.example.com:3389 \
  --type PingDirectoryProxy-server \
  --set server-host-name:intra-proxy-host \
  --set server-port:338 \
  --set "bind-dn:cn=Directory Manager" \
  --set "password:secret123"
```

2. Create a global attribute index on the `uid` attribute as follows:

```
$ bin/dsconfig create-global-attribute-index \
  --processor-name dc_example+dc+com-eb-req-processor \
  --index-name uid \
```

3. Change the entry-balancing request processor to load the indexes at startup from the peer Directory Proxy Server using `dsconfig set-request-processor-prop` as described above.

To Persist the Global Index from a File

The PingDirectoryProxy Server supports periodically persisting the global index to a file and priming the global index from the persisted file when the server is restarted.

An Entry Balancing Request Processor can be configured to periodically persist the global index to disk, so that when the Entry Balancing Request Processor is reinitialized (on startup), it can prime the values from disk instead of putting load on the remote servers. Being able to read the index from disk eliminates the load on backend Directory Server instances if many PingDirectoryProxy Server instances were to come up at once.

An entry-balancing request processor can be configured to persist the global index to disk by including `file` as one of the prime index sources (with the `prime-index-source` property). The frequency at which the file is written is controlled by the `persist-global-index-frequency` property.

The global index needs to be fully primed before it will be persisted. It can be initially primed using a peer PingDirectoryProxy Server or from a backend Directory Server. On a running PingDirectoryProxy Server, when new global attribute indexes are added, the global index can be primed with those attribute indexes by running the `rebuild-index` tool. The `rebuild-index` tool always uses a remote server for priming the global index even if `file` is configured as a source). On subsequent restarts of the PingDirectoryProxy Server, the global index will be primed from the persisted file instead of going over the network to a remote server, which allows it to be primed much faster than if it were using a remote priming source. Also, during server startup, the global index priming works by using each configured `prime-index-source` property in the specified order until it is fully primed to take advantage of what is available locally before contacting one or more remote servers.

- The following `dsconfig` command prime all indexes at startup from a file.

```
dsconfig -n set-request-processor-prop \
  --processor-name entry-balancing \
  --set prime-index-source:file \
  --set prime-index-source:ds \
  --set persist-global-index-frequency:10s \
  --set persist-global-index-directory:/servers/proxy-1/index-files \
  --set prime-all-indexes:true
```

Priming or Reloading the Global Indexes from Sun Directory Servers

When priming or reloading a global index based on a Sun Directory Server environment, the Sun servers may become overwhelmed and unresponsive because of their method of streaming data. To reduce the impact of priming on these server, you can use the `prime-search-entry-per-second` property. To reduce the impact of reloading these indexes, use the `--searchEntryPerSecond` property of the `reload-index` command. These properties control the rate at which the Directory Proxy Server accepts search result entries from the backend directory servers.

To find the optimum rate, start low and specify a few thousand search entries per second. Then increase as necessary.

Working with Alternate Authorization Identities

Access control rules in an entry-balanced deployment are configured in the Directory Server backend servers and require access to the entry contents of the user *issuing* the request. This can introduce a possible issue when clients to the Directory Proxy Server authenticate as users whose entries are among the entry-balanced sets. If the server which is processing a request does not contain the issuing user's entry, then the access control cannot be evaluated.

For example, consider a deployment that has two entry-balancing sets, set-01 and set-02. Set-01 has entries in the range `uid=0-10000`, while set-02 has entries for `uid=10001-20000`. The client with `uid=5000` binds to the Directory Proxy Server, which sends a BIND request to entry-balancing set-01. Next, the client sends a SEARCH request with filter "`(uid=15000)`". The Directory Proxy Server determines that `uid=15000` lives on entry-balancing set-02. The Directory Proxy Server then determines that the entry for the authenticated user with `uid=5000` does not exist in set-02 and that the access control handler would reject the SEARCH request issued by an unknown user.

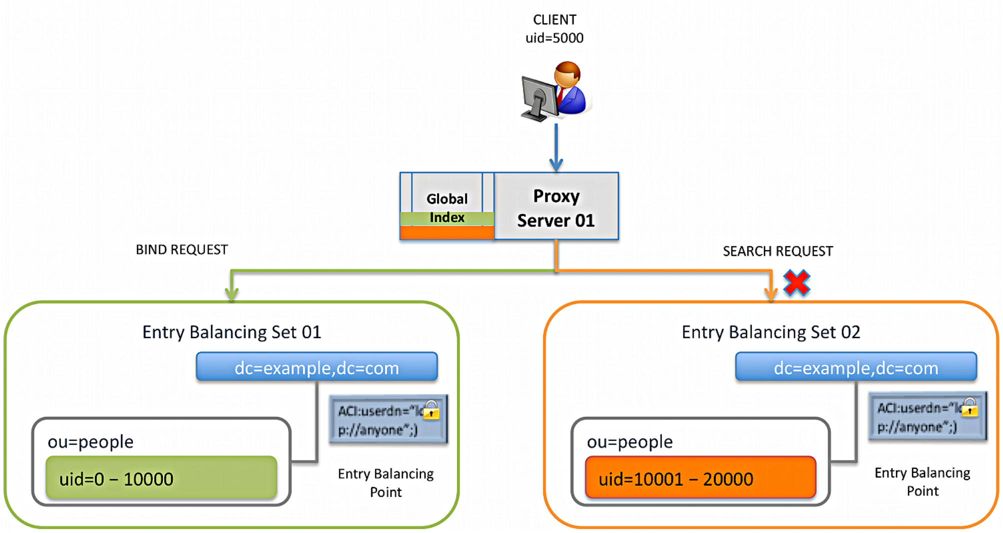


Figure 5: Entry-Balancing Issue with Clients Not Present in the Underlying Data Set

One solution to this problem is to make use of an *alternate authorization identity* for the user, which references an entry that exists in all Directory Servers in all backend sets and has an equivalent set of access control rights as the authenticated user. The alternate authorization identity is used when the Directory Proxy Server observes that the Directory Server processing a request does not contain the entry of the user issuing the request.

The following sections cover the procedures to configure the alternate authorization identities for the Directory Proxy Server.

About Alternate Authorization Identities

Whenever the Directory Proxy Server forwards a request to the backend set containing the user's entry, it forwards the request with an authorization identity that reflects the user's actual identity, since servers in that set already know about that user. However, when forwarding a request to a backend set that does *not* contain the user's entry, the Directory Proxy Server uses an *alternate authorization identity* that reflects the generic user with the same set of rights as the actual user issuing the request. Alternate authorization identities allow for the proper evaluation of access control rules for users whose entries are not present within an entry-balanced dataset.

There are typically only a few different generic class of users from an access control perspective, which can be placed in a portion of the DIT that is not below the entry-balancing base DN and is replicated to all servers in the topology. For example, assume that you have three classes of users: full administrators, password administrators, and normal users. You could create the following entries in the topology and assign them the appropriate access rights:

uid=normal user,dc=example,dc=com
uid=server-admin,dc=example,dc=com
uid=password-admin,dc=example,dc=com

Returning to the example scenario, the client with `uid=5000` binds to the Directory Proxy Server, which sends a BIND request to entry-balancing set-01. Next, the client sends a SEARCH request for `uid=15000`. The Directory Proxy Server determines that `uid=15000` lives on entry-balancing set-02. Next, the Directory Proxy Server then determines that the client `uid=5000` does not have an entry on entry-balancing set-02. The Directory Proxy Server uses an *alternate authorization identity* that reflects the generic user, `uid=normal user`, which has the same set of rights as the client `uid=5000` who is issuing the request. The access control is accepted and the SEARCH request returns a response for `uid=5000`.

Whenever a user authenticates to the Directory Proxy Server, the server can keep track of which backend set holds that user's entry and determine whether an alternate authorization identity is required. The server can also determine which of these generic accounts best describes the rights that the user should have.

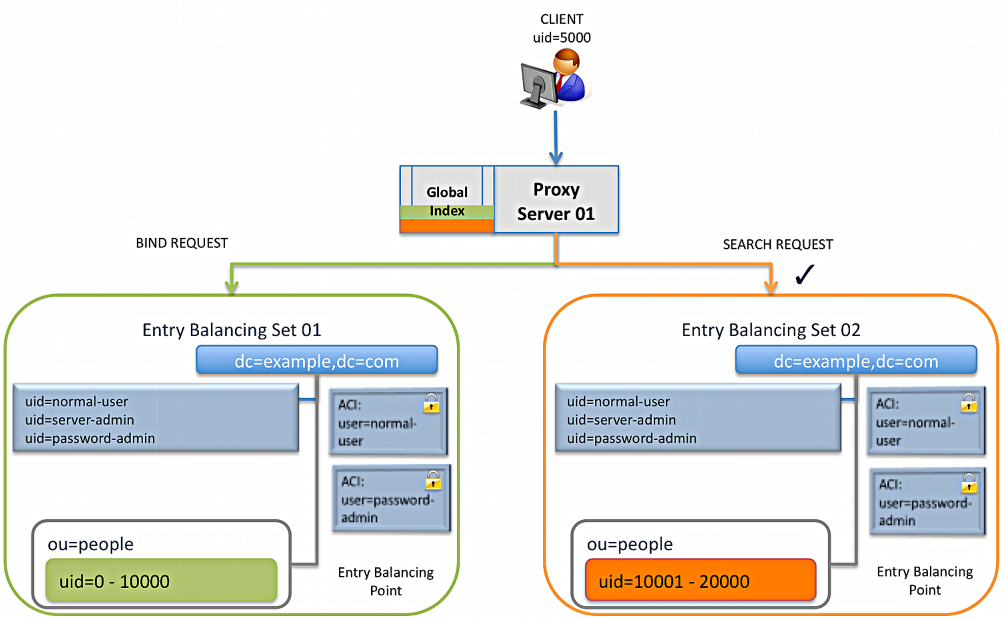


Figure 6: Alternate Authorization Identity Solves Access Control Issues in Entry-Balancing Deployments

When an alternate authorization identity is invoked, you will see authzID='dn:uid=normal user,dc=example,dc=com' in the server log, indicating that the alternate authorization identity was used. For example, if the user.15000 is in a different backend set from user.5000, the log will show the following:

```
% bin/ldapsearch -D "uid=user.5000,ou=people,dc=example,dc=com" -w pasword \
  -b uid=user15000,ou=people,dc=example,dc=com "(objectclass=*)"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2
via="app='Directory-Proxy address='127.0.0.1'
authzID='dn:uid=normal user,dc=example,dcom' sessionID='conn=2'
requestID='op=1'" base="uid=user.150000,ou=people,dc=example,dc=com"scope=2
filter="(objectclass=*)" attrs="ALL"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2 resultCode=0
 etime=2.038
entriesReturned=1 authzDN="uid=normal-user,dc=example,dc=com"
```

Configuring Alternate Authorization Identities

Alternate authorization identities are specified by the authz-attribute property of the entry-balancing request processor configuration object. By default, the authz-attribute property has the default value of ds-authz-map-to-dn, which is an attribute reserved for this purpose.

To Configure Alternate Authorization Identity DNs

If a user entry has a value for ds-authz-map-to-dn whether it's explicitly contained in the entry or only present via a virtual attribute, then that will be used to specify the alternate authorization identity for the user. Otherwise, the default authorization identity (as indicated via the authz-dn configuration property) will be used to determine the alternate authorization identity.

1. Use dsconfig to set the authz-dn property of the entry-balancing request processor configuration. If any user among the balanced entries does not have an alternate authorization identity defined, the Directory Proxy Server will use the value of the authz-dn property of the entry-balancing request processor configuration.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set "authz-dn:uid=normal user,dc=example,dc=com"
```

2. Create an auxiliary object class containing `ds-authz-map-to-dn` as an allowed attribute.
3. Add the auxiliary object class value to all user entries of interest.
4. Then, add the following attribute value to a `server-admin` user.

```
ds-authz-map-to-dn: uid=server-admin,dc=example,dc=com
```

Chapter

7

Managing Entry-Balancing Replication

Topics:

Replication in the PingDirectoryProxy Server synchronizes directory data between all servers in the topology. In a deployment using the entry-balancing feature, however, directory data under the entry-balancing point is split into multiple data sets. Each data set is replicated to ensure high availability between a subset of the servers in the topology. Other directory data, such as the schema or data above the entry-balancing point, is replicated between all servers in the topology.

This chapter presents the following information about replication in an entry-balancing environment:

Overview of Replication in an Entry-Balancing Environment

In an entry-balanced deployment, some data is replicated everywhere, such as the schema, the server registry, and other shared data, and some data is replicated only on certain servers. A replication domain contains all of the servers in a replicated topology and shares a schema. The replication domain is associated with the base DN and must be a base DN of a backend.

By default, replication propagates updates to all replication servers in the topology. Updates to data under the entry-balancing point, however, must be replicated only among server instances in the same data set. Replication requires that, in such deployments, the Directory Server is configured with a replication set name global configuration property, and two backends. One backend has a base DN that is replicated globally (such as `dc=example,dc=com`) and the second backend has a base DN associated with the entry-balancing point (such as `ou=people,dc=example,dc=com`).

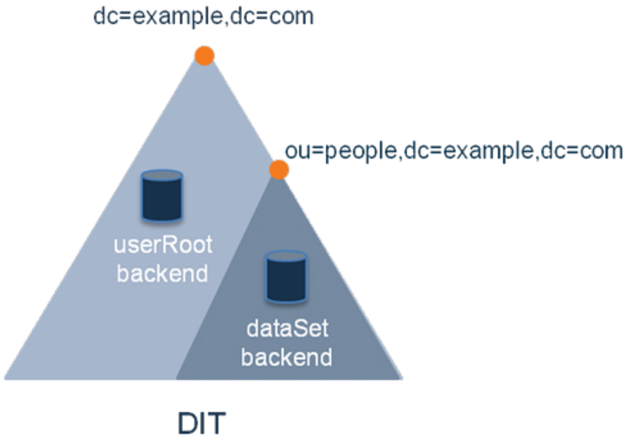


Figure 7: Global and Restricted Backends

If a data set name is not defined when you set up the Directory Proxy Server, one will be provided by default. The proper configuration of an entry-balancing environment requires coordination between the Directory Server and Directory Proxy Server. Once replication is enabled, the replication domain may be designated as the domain participating in entry balancing.

Review the *PingData Directory Server Administration Guide* for more details about replication, managing the replication topology, and working with multiple backends.

Replication Prerequisites in an Entry-Balancing Deployment

Replication in an entry-balanced deployment requires the following:

- **Multiple local DB backends**. When you set up the Directory Server instances, you need two backends, a global backend for globally replicated data, such as `userRoot`, and a backend for the balancing point base DN, `dataSet`. Both backends need to be enabled for replication and initialized separately.
- **Replication set name**. Every Directory Server in your replicated topology must have a replication set name. This replication set name coordinates the Directory Proxy Server and the Directory Server. The restricted domain is only replicated within instances using the same replication set name.
- **Multiple Directory Proxy Server subtree views**. The entry-balanced proxy configuration relies on multiple subtree views, one for the globally replicated base DN and one for the entry-balancing point base DN. The globally replicated base DN will have a proxying request processor associated with it. The restricted base DN will have an entry-balancing request processor associated with it. This configuration is best achieved using the `create-initial-proxy-config` tool after running setup.

About the --restricted Argument of the dsreplication Command-Line Tool

When enabling replication for a server that takes part in an entry balanced environment, it is recommended that the multiple domains involved are enabled at the same time. There is a global domain, and a restricted domain, where the restricted domain represents the entry-balancing point. Each base DN is defined in a separate Local DB Backend. The dsreplication CLI tool has a --restricted argument that is used to specify which base DN is considered an entry-balancing point.

To Use the --restricted Argument of the dsreplication Command-Line Tool

- Run dsreplication to enable replication between two servers with entry balancing.
  - You can run the command in non-interactive mode as follows:

```
$ bin/dsreplication enable --host1 host1.example.com \
  --port1 1389 --bindDN1 "cn=Directory Manager" \
  --bindPassword1 secret --replicationPort1 8989 \
  --host2 host2.example.com --port2 2389 \
  --bindDN2 "cn=Directory Manager" --bindPassword2 secret \
  --replicationPort2 8989 --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --restricted ou=people,dc=example,dc=com
```

  - Alternatively, you can enable replication using the interactive command line, making sure to specify that an entry balancing is being used and specifying the base DN of the entry-balancing point. After entering dsreplication and entering the LDAP connection parameters, follow the prompts presented.

```
You must choose at least one base DN to be replicated.

Replicate base DN dc=example,dc=com? (yes / no) [yes]: yes

Replicate base DN ou=people,dc=example,dc=com? (yes / no) [yes]: yes

Do you plan to configure entry balancing using the Directory Proxy Server?
 (yes / no) [no]: yes

Is dc=example,dc=com an entry-balancing point? (yes / no) [no]: no

Is ou=people,dc=example,dc=com an entry-balancing point? (yes / no) [no]:
 yes
```

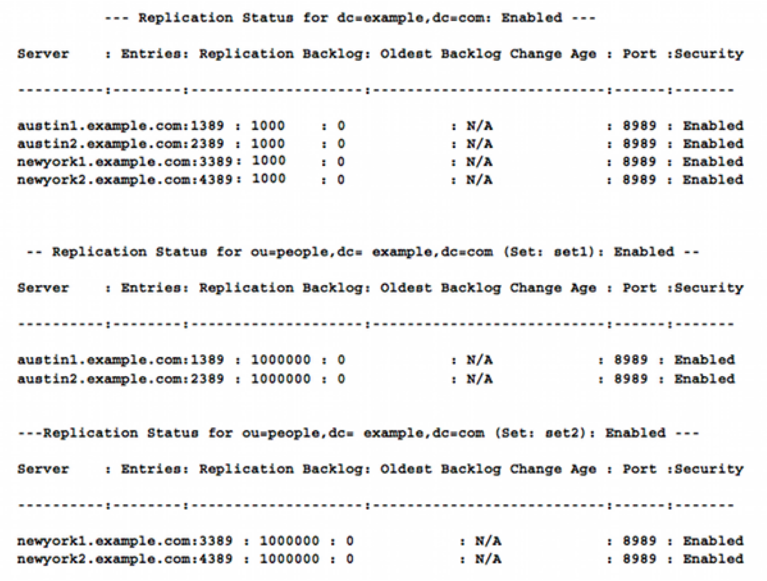
Checking the Status of Replication in an Entry-Balancing Deployment

You can use the dsreplication status tool to check the status of an entry-balancing deployment. In this example, the ou=people,dc=example,dc=com subtree is entry-balanced. The data is split into two sets, set1 and set2. The servers host1 and host2 are in replication set set1 and servers host3 and host4 are in replication set set2.

To Check the Status of Replication in an Entry-Balancing Deployment

- Run the dsreplication command to get a status of replication in the entry-balancing deployment. To view a specific set, use the --setName option to see only the specific replication set; otherwise, all of the sets will be displayed by default.

```
$ bin/dsreplication status --hostname host1.example.com \
  --port 1389 --adminUID admin --adminPassword secret
```



```
          --- Replication Status for dc=example,dc=com: Enabled ---

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:-------------------:--------------------------:------:-------

austin1.example.com:1389 : 1000    : 0            : N/A              : 8989 : Enabled
austin2.example.com:2389 : 1000    : 0            : N/A              : 8989 : Enabled
newyork1.example.com:3389: 1000    : 0            : N/A              : 8989 : Enabled
newyork2.example.com:4389: 1000    : 0            : N/A              : 8989 : Enabled


 -- Replication Status for ou=people,dc= example,dc=com (Set: set1): Enabled --

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:-------------------:--------------------------:------:-------

austin1.example.com:1389 : 1000000 : 0            : N/A             : 8989 : Enabled
austin2.example.com:2389 : 1000000 : 0            : N/A             : 8989 : Enabled

---Replication Status for ou=people,dc= example,dc=com (Set: set2): Enabled ---

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:-------------------:--------------------------:------:-------

newyork1.example.com:3389 : 1000000 : 0           : N/A             : 8989 : Enabled
newyork2.example.com:4389 : 1000000 : 0           : N/A             : 8989 : Enabled
```

Example of Configuring Entry-Balancing Replication

This section describes how to set up a four-server replication topology that uses entry balancing to distribute entries across the servers. The procedure assumes that none of the servers have participated in any previous replication topology. This is supported for one or multiple entry balancing domains.

Assumptions

The example uses the LDAP (389) and replication (8989) ports respectively. It configures the following hosts:

austin1.example.com
newyork1.example.com
austin2.example.com
newyork2.example.com

In this example, we have a global domain of `dc=example,dc=com`, which is replicated across all servers. The data below the entry-balancing point of `ou=people,dc=example,dc=com` is distributed across two data sets, dataSet1 and dataSet2. Each data set is replicated between two directory servers. Each of these servers is associated with one of two locations, Austin and New York.

Configuration Summary

To configure replication in an entry-balanced deployment, you must do the following:

- Install two directory servers in an Austin location and two directory servers in a New York location.
- Create a new backend, called `dataset`, to store the entry-balancing data set.
- Define entry-balancing set names `dataSet1` and `dataSet2` for assignment to the `replication-set-name` Global Configuration Property of the Directory Server instances.
- Import the data representing the global domain, stored in userRoot, into a server. Choose a server for each of the entry-balancing data sets, both stored in the backend named `dataset`.
- Enable replication and initialize remaining servers.
- Configure the proxies.
- Check the status of replication.

To Install the Directory Server

First, install the Directory Server instances. In this example, we install the following four servers, two in the Austin location and two in the New York location:

austin1.example.com
austin2.example.com
newyork1.example.com
newyork2.example.com

1. We install the first server, austin1, as follows:

```
root@austin1# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

2. Install the second Austin server, austin2, in the same way:

```
root@austin2 # ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

3. Next, install the two New York servers, newyork1 and newyork2, as follows:

```
root@newyork1# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense

root@newyork# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

To Create the Database Backends and Define the Replication Set Name

1. On all servers, create the dataset backend as follows:

```
./bin/dsconfig --no-prompt create-backend \
--backend-name dataset --type local-db --set enabled:true \
--set base-dn:ou=people,dc=example,dc=com
```

2. Set the replication set name for `austin1.example.com` and `newyork1.example.com` to dataset1:

```
./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset1
```

3. Set the replication set name for `austin2.example.com` and `newyork1.example.com`to dataset2:

```
./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset2
```

To Create and Set the Locations

1. On the Austin servers, create the two locations, newyork and austin, and set the location of this instance to austin:

```
./bin/dsconfig --no-prompt create-location --location-name austin

./bin/dsconfig --no-prompt create-location --location-name newyork \
--set preferred-failover-location:austin

./bin/dsconfig --no-prompt set-location-prop --location-name austin \
--add preferred-failover-location:newyork

./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:austin
```

2. For the New York servers, set the location to newyork:

```
./bin/dsconfig --no-prompt create-location \
--location-name austin

./bin/dsconfig --no-prompt create-location \
--location-name newyork \
--set preferred-failover-location:austin

./bin/dsconfig --no-prompt set-location-prop \
--location-name austin \
--add preferred-failover-location:newyork

./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:newyork
```

To Import the Entries

We import the userRoot data, based on data defined in the `userRoot.ldif` file, into one server. This file does not contain entries at or within the entry-balancing point, `ou=people,dc=example,dc=com`.

1. Use the `import-ldif` command to import the userRoot data.

```
root@austin1# ./bin/import-ldif --backendID userRoot \
--ldifFile /data/userRoot.ldif \
--includeBranch dc=example,dc=com \
--rejectFile /data/austin1-import-rejects \
--port 389
--hostname austin1.example.com
```

2. Import the dataSet1 data on one server into the dataset backend, which is assigned the dataSet1 `replication-set-name`.

```
root@austin1# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset1.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/austin1-dataset-import-rejects \
--hostname austin1.example.com --port 389
```

3. Import the dataSet2 data on one server into the dataset backend, which is assigned the dataSet2 `replication-set-name`.

```
root@austin2# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset2.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/austin2-dataset-import-rejects \
--hostname austin2.example.com --port 389
```

To Enable Replication in an Entry-Balancing Deployment

Now we can enable replication between the servers and initialize the remaining servers without data. Notice that we specify the `--restricted` domain in the `dsreplication` command.

1. Run `dsreplication enable` to enable the servers in the topology. The first invocation of this command creates the admin account.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 austin2.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

2. Enable replication between austin1 and newyork1. This procedure automatically enables replication between austin2 and newyork1 as well.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 newyork1.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

3. Enable replication between austin1 and newyork2. This will complete the entry-balancing replication setup.

```
root@austin1# ./bin/dsreplication enable \
  --host1 austin1.example.com \
  --port1 389 --bindDN1 "cn=directory manager" \
  --bindPassword1 pass --host2 newyork2.example.com \
  --port2 389 --bindDN2 "cn=directory manager" \
  --bindPassword2 pass \
  --replicationPort1 8989 \
  --replicationPort2 8989 \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --restricted ou=people,dc=example,dc=com \
  --adminUID admin --adminPassword pass --trustAll \
  --no-prompt
```

4. Initialize the remaining servers without data. The global domain, `dc=example,dc=com` needs to be initialized on austin2, newyork1 and newyork2. The `ou=people,dc=example,dc=com` entry-balancing domain needs to be initialized from austin1 to newyork2, and then again from austin2 to newyork2. We will combine these steps by initializing both domains with one invocation once austin2 is initialized with the global domain.

```
root@austin1# ./bin/dsreplication initialize \
  --hostSource austin1.example.com --portSource 389 \
  --hostDestination austin2.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --no-prompt

root@austin1# ./bin/dsreplication initialize \
  --hostSource austin1.example.com --portSource 389 \
  --hostDestination newyork1.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --no-prompt

root@austin2# ./bin/dsreplication initialize \
  --hostSource austin2.example.com --portSource 389 \
  --hostDestination newyork2.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --no-prompt
```

To Check the Status of Replication

Once replication has been configured, check the status of the replication topology using the `dsreplication status` command.

- Run the `dsreplication status` command to check its status.

```
root@austin1# ./bin/dsreplication status \
--adminPassword pass --no-prompt --port 389
```

Chapter 8

Managing the Directory Proxy Server

Topics:

Once you have configured the PingDirectoryProxy Server, you can manage the day-to-day operations of your deployment using the monitoring and logging features. This chapter provides procedures to help you configure logging and monitor your deployment.

This chapter includes the following sections:

Managing Logs

The Directory Proxy Server provides a number of different types of log publishers that can be used to provide information about how the server is processing.

About the Default Logs

You can view all logs in the `PingDirectoryProxy/logs` directory. This section provides information about the following default logs:

Error Log
server.out Log
Debug Log
Config Audit Log and the Configuration Archive
Access Log
Setup Log
Tool Log
LDAP SDK Debug Log

Error Log

By default, this log file is available at `logs/errors` below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE msgID=458887
 msg="The Directory Server has started successfully"
```

The category field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL, ADMIN, ADMIN_TOOL, BACKEND, CONFIG, CORE, DSCONFIG, EXTENSIONS, PROTOCOL, SCHEMA, JEB, SYNC, LOG, PLUGIN, PROXY, QUICKSETUP, REPLICATION, RUNTIME_INFORMATION, TASK, THIRD_PARTY, TOOLS, USER_DEFINED, UTIL, VERSION.

The severity field provides information about how severe the server considers the problem to be. Available severities include:

- **DEBUG** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Proxy Server provides a separate debug logging facility as described below.
- **INFORMATION** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.
- **MILD_WARNING** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.
- **MILD_ERROR** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.
- **NOTICE** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.
- **SEVERE_WARNING** – Used for problems that the server detects that might lead to bigger problems in the future and should be addressed by administrators.
- **SEVERE_ERROR** – Used for significant problems that have prevented the server from successfully completing processing and are considered important.

- **FATAL_ERROR** – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the NOTICE, SEVERE_WARNING, SEVERE_ERROR, and FATAL_ERROR severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the PROTOCOL category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-
warning,severe-error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the DEBUG severity for a category will not automatically enable logging at the INFORMATION, MILD_WARNING, or MILD_ERROR severities.

The error log configuration may be altered on the fly using tools like `dsconfig`, the Administrative Console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out Log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. The `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug Log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- **enabled** – Indicates whether debug logging is enabled. By default, it is disabled.
- **log-file** – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- **default-debug-level** – Specifies the minimum log level for debug messages that should be written. The default value is “error,” which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include warning, info, and verbose. Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the info debug level automatically enables the warning and error levels.
- **default-debug-category** – Specifies the categories for debug messages that should be written. Some of the most useful categories include caught (provides information and stack traces for any exceptions caught during processing), database-access (provides information about operations performed in the underlying database), protocol (provides information about ASN.1 and LDAP communication performed by the server), and data (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files to help isolate information that might be relevant to a particular problem.

Note: Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and then be scoped so that only pertinent debug information is recorded.

Debug targets can be used to further pare down the set of messages generated. For example, you can specify that the debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and interpret the output.

Audit log

The audit log is a specialized version of the access log, used for troubleshooting problems that may occur in the course of processing. The log records all changes to directory data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

By default, if you enable the audit log on the Directory Proxy Server, the `userPassword` and `authPassword` attribute values are obscured, Each value of an obscured attribute is replaced in the audit log with a string of the form `"***** OBSCURED VALUE *****"`. You can unobscure these attributes by deleting them from the `obscure-attribute` property.

Config Audit Log and the Configuration Archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the `logs/config-audit.log` file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the `dsconfig` command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.
- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Proxy Server maintains complete copies of all previous configurations. These configurations are provided in the `config/archived-configs` directory and are gzip-compressed copies of the `config/config.ldif` file in use before the configuration change was made. The filenames contain time stamps that indicate when that configuration was first used.

Access and Audit Log

The access log provides information about operations processed within the server. The default access log file is written to `logs/access`, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1"
 to="127.0.0.1"
  protocol="LDAP+TLS" clientConnectionPolicy="default"
```

```
[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
  dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
 clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2
  base="ou=People,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL"
  resultCode=0 etime=0.549 entriesReturned=1
[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensi- tive or not useful.

Result messages include a `resultCode` element that indicates whether the operation was successful or if failed and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **origin=replication** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **message** – Text that was included in the `diagnosticMessage` field of the response sent to the client.
- **additionalInfo** – Additional information about the operation that was not included in the response sent back to the client.
- **authDN – DN** of the user that authenticated to the server (typically only included in bind result messages).
- **authzDN – DN** of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **authFailureID** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).
- **authFailureReason** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **responseOID** – OID included in an extended response returned to the client.
- **entriesReturned** – Number of matching entries returned to the client for a search operation.
- **unindexed=true** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The UnboundID LDAP SDK for Java provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Proxy Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify DN) processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to `logs/audit`) in LDIF form.

The PingDirectoryProxy Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup Log

The `setup` tool writes a log file providing information about the processing that it performs. By default, this log file is written to `logs/setup.log` although a different name may be used if a file with that name already exists, because the `setup` tool has already been run. The full path to the setup log file is provided when the `setup` tool has completed.

Tool Log

Many of the administrative tools provided with the Directory Proxy Server (for example, `import-ldif`, `export-ldif`, `backup`, `restore`, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the `logs/tools` directory (for example, **logs/tools/ import-ldif.log**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

LDAP SDK Debug Log

This log can be used to help examine the communication between the Directory Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

Types of Log Publishers

The PingDirectoryProxy Server provides a number of differently types of loggers that can be used to get processing information about the server. There are three primary types of loggers:

- **Access loggers** provide information about operations processed within the server. They can be used for understanding the operations performed by clients and debugging problems with directory-enabled applications, and they can also be used for collecting usage information for performance and capacity planning purposes.
- **Error loggers** provide information about warnings, errors, or significant events that occur within the server.
- **Debug loggers** can provide detailed information about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database.

By default, the following log publishers are enabled on the system:

File-based access logger
File-based error logger
Failed-operations access logger

The PingDirectoryProxy Server also provides the follow log publishers that are disabled by default:

File-based debug logger
File-based audit logger
Expensive operations access logger
Successful searches with no entries returned access logger

Creating New Log Publishers

The PingDirectoryProxy Server provides customization options to help you create your own log publishers with the `dsconfig` command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see Configuring Log Rotation and Configuring Log Retention.

To Create a New Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

Note: To configure compression on the logger, add the option to the previous command:

```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. If needed, view log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. On the main menu, select the option to configure the log publisher.
4. On the **Log Publisher** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the Directory Proxy Server root. For example, `logs/disconnect.log`.
9. Select the rotation policy to use for this log publisher.
10. Select the retention policy to use for this log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Type `f` to apply the changes.

About Log Compression

The Directory Proxy Server supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The `summarize-access-log` tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting purposes, administrators using compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the `compression-mechanism` property to have the value of "gzip" when creating a new logger.

About Log Signing

The Directory Proxy Server supports the ability to cryptographically sign a log to ensure that it has not been modified in any way. For example, financial institutions require audit logs for all transactions to check for correctness. Tamper-proof files are therefore needed to ensure that these transactions can be propertly validated and ensure that they have not been modified by any third-party entity or internally by unscrupulous employees. You can use the `dsconfig` tool to enable the `sign-log` property on a Log Publisher to turn on cryptographic signing.

When enabling signing for a logger that already exists and was enabled without signing, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the `validate-file-signature` tool provided in the `bin` directory of the server (or the `bat` directory for Windows systems).

Once you have enabled this property, you must disable and then re-enable the Log Publisher for the changes to take effect.

About Encrypting Log Files

The Directory Proxy Server supports the ability to encrypt log files as they are written. The `encrypt-log` configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an `.encrypted` extension (and if both encryption and compression are enabled, the extension will be `.gz.encrypted`). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The `encrypt-file` tool (or custom code, using the LDAP SDK's `com.unboundid.util.PassphraseEncryptedInputStream`) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the `encryption-settings create` command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the `encryption-settings-definition-id` property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the `rotate-log` tool to force the file to be rotated before attempting to examine it.

To Configure Log Signing

1. Use `dsconfig` to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set enabled:false
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set enabled:true
```

To Validate a Signed File

The Directory Proxy Server provides a tool, `validate-file-signature`, that checks if a file has not been tampered with in any way.

- Run the `validate-file-signature` tool to check if a signed file has been tampered with. For this example, assume that the `sign-log` property was enabled for the File-Based Audit Log Publisher.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

Note: If any validations errors occur, you will see a message similar to the one as follows:

```
One or more signature validation errors were encountered
while validating the contents of file 'logs/audit':
* The end of the input stream was encountered without
  encountering the end of an active signature block.
  The contents of this signed block cannot be trusted
  because the signature cannot be verified
```

To Configure Log File Encryption

1. Use `dsconfig` to enable encryption for a Log Publisher. In this example, the File-based Access Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted Access"
 \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-definition-
id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...Done
Writing decrypted data to file '/ds/PingDirectory/decrypted-access' using a
```

```
key generated from encryption settings definition
 '332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'
Successfully wrote 123,456,789 bytes of decrypted data
```

Configuring Log Rotation

The Directory Proxy Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Proxy Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

To Configure the Log Rotation Policy

- Use `dsconfig` to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring Log Rotation Listeners

The Directory Proxy Server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed. Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true
```

The path specified by the `copy-to-directory` property must exist, and the filesystem containing that directory must have enough space to hold all of the log files that will be written there. The server will automatically monitor free disk space on the target filesystem and will generate administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the `summarize-access-log` tool on a recently-rotated log file and writes its output to a file in a specified location. This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the `summarize-access-log` tool, including the `file-based-access` and `operation-timing-access` logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of `.summary`. If the `output-directory` property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically be deleted. Though files are generally small in comparison to the log files themselves, make sure that there is enough space available in the specified storage directory. The server automatically monitors free disk space on the filesystem to which the summary files are written.

Configuring Log Retention

The Directory Proxy Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Proxy Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Directory Proxy Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Custom Retention Policy**. Create a new retention policy that meets your Directory Proxy Server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

- Use `dsconfig` to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Setting Resource Limits

You can set resource limits for the Directory Proxy Server using several global configuration properties as well as setting resource limits on specific client connection policies. If you configure both global and client connection policy resource limits, the first limit reached will always be honored. For example, if the server-wide maximum concurrent connections limit is reached, then all subsequent connection will be rejected until existing connections are closed, regardless of whether a client connection policy limit has been reached.

Setting Global Resource Limits

You can specify the following types of global resource limits:

- Specify the maximum number of client connections that can be established at any given time using the `maximum-concurrent-connections` property. If the server already has the maximum number of connections established, then any new connection attempts from any clients will be rejected until an existing connection is closed. The default value of zero indicates that no limit is enforced.

- Specify the maximum number of client connections that can be established at any give time from the same client system using the `maximum-concurrent-connections-per-ip-address` property. If the server already has the maximum number of connections established from a given client, then any new connection attempts from that client will be rejected until an existing connection from that client is closed. The server may continue to accept connections from other clients that have not yet reached this limit. The default value of zero indicates that no limit is enforced.
- Specify the maximum number of client connections that can be established at any given time while authenticated as a particular user with the `maximum-concurrent-connections-per-bind-dn` property. This property applies after the connection is established, because the bind operation to authenticate the user happens after the connection is established rather than during the course of establishing the connection itself. If the maximum number of connections are authenticated as a given user, then any new attempt to authenticate as that user will cause the connection performing the bind to be terminated. Note that this limit applies only to authenticated connections, and will not be enforced for clients that have not authenticated or for clients that have authenticated as the anonymous user. The default value of zero indicates that no limit is enforced.

Any changes to the `maximum-concurrent-connections` and `maximum-concurrent-connections-per-ip-address` properties will take effect only for new connections established after the change is made. Any change to the `maximum-concurrent-connections-per-bind-dn` property will apply only to connections (including existing connections) which perform authentication after the change is made. Existing connections will be allowed to remain established even if that would cause the new limit to be exceeded.

Setting Client Connection Policy Resource Limits

You can also configure resource limits in a client connection policy using the following properties of the client connection policy:

- **maximum-concurrent-connections**. This property specifies the maximum number of client connections that may be associated with a specific client connection policy at any given time. Once this limit has been reached, any further attempts to associate a connection with this client connection policy will result in the termination of the connection.
- **maximum-connection-duration**. This property specifies the maximum length of time that a connection associated with a particular client connection policy may be established. When the connection has been established longer than this period, it will be terminated.
- **maximum-idle-connection-duration**. This property specifies the maximum time that a connection associated with a particular client connection policy may remain established after the completion of the last operation processed on that connection. Any new operation requested on the connection resets the timer. Connections that are idle for longer than the specified time will be terminated.
- **maximum-operation-count-per-connection**. This property specifies the maximum number of operations that may be requested by any client connection associated with this client connection policy. If an attempt is made to process more than this number of operations on the connection, then the connection will be terminated.
- **maximum-concurrent-operations-per-connection**. This property specifies the maximum number of concurrent operations that can be in progress for any connection. This property can be used to prevent a single client connection from monopolizing server processing resources by sending a large number of concurrent asynchronous requests.
- **maximum-connection-operation-rate**. This property specifies the maximum rate at which a client associated with a specific client connection policy may issue requests to the Directory Proxy Server. If a client attempts to request operations at a rate higher than this limit, then the server will behave as described by the `connection-operation-rate-exceeded-behavior` property.
- **connection-operation-rate-exceeded-behavior**. This property describes how the server should behave if a client connection attempts to exceed a rate defined in the `maximum-connection-operation-rate` property.
- **maximum-policy-operation-rate**. This property specifies the maximum rate at which all clients associated with a particular client connection policy may issue requests to the Directory Proxy Server. If this limit is exceeded, then the server will exhibit the behavior described in the `policy-operation-rate-exceeded-behavior` property.
- **policy-operation-rate-exceeded-behavior**. This property specifies the behavior of the Directory Proxy Server if a client connection attempts to exceed the rate defined in the `maximum-policy-operation-rate` property.

Monitoring the Directory Proxy Server

While the Directory Proxy Server is running, it generates a significant amount of information available through monitor entries. This section contains information about the following:

- Monitoring Server Status Using the status Tool
- About the Monitor Entries
- Using the Monitoring Interfaces
- Monitoring with JMX

Monitoring System Data Using the Data Metrics Server

The PingData Data Metrics Server provides collection and storage of performance data from your server topology. You can use the System Utilization Monitor with the Data Metrics Server to collect information about the host system CPU, disk, and network utilization on any platform except Linux. If you are not using the Data Metrics Server, you do not need to use the system utilization monitor. When data is being collected, it periodically forks the process and executes commands.

For more information about using the System Utilization Monitor, refer to the data collection chapter of the PingData Data Metrics Server documentation.

To Monitor Server Using the Status Tool

The PingDirectoryProxy Server provides a `status` tool that provides basic server status information, including version, connection handlers, a table of LDAP external servers, and the percent of the global index that is used.

1. Run the `status` tool to view the current state of the server.

```
$ bin/status
```

2. Enter the LDAP connection parameters.

```
>>>> Specify LDAP connection parameters

Administrator user bind DN [cn=Directory Manager]:

Password for user ‘cn=Directory Manager’:
```

```
          --- Server Status ---
Server Run Status:    Started 07/Jan/2011:10:59:52.000 -0600
Operational Status:   Available
Open Connections:     4
Max Connections:      8
Total Connections:    25

          --- Server Details ---
Host Name:            example
Administrative Users: cn=Directory Manager
Installation Path:    /path/to/PingDirectoryProxy
Version:              PingDirectoryProxy Server 7.0.0.0
Java Version:         jdk-7u9

          --- Connection Handlers ---

Address:Port : Protocol : State
-------------:----------:---------

0.0.0.0:1689 : JMX      : Disabled
0.0.0.0:636  : LDAPS    : Disabled
0.0.0.0:9389 : LDAP     : Enabled
```

```
         --- LDAP External Servers ---

Server         : Status    : Score : LB Algorithm
---------------:-----------:-------:----------------------
localhost:389  : Available : 10    : dc_example_dc_com-failover
localhost:1389 : Available : 10    : dc_example_dc_com-failover


         --- LDAP External Server Op Counts ---

Server         : Add : Bind:Compare:Delete:Modify:Mod DN:Search : All
---------------:-----:----:------:-------:-------:------:-------:----
localhost:11389: 0   : 0  : 0    : 0     : 0     : 0    : 1249  : 1249
localhost:12389: 0   : 0  : 0    : 0     : 0     : 0    : 494   : 494



         --- Entry Balancing Request Processors ---

 Base DN                     : Global Index % Used
----------------------------:--------------------
ou=people,dc=example,dc=com : 33


         --- Global Index Stats for ou=people,dc=example,dc=com ---

Index : Total Bytes : Key Bytes : Keys : Size (# Keys) : Inserted :
Removed : Replaced: Hits : Misses : Discarded : Duplicates
------:-------------:-----------:---------:---------------:----------
rdn   : 30667304    : 14888906  : 1000001 : 3464494 0 :0 :0 :0 :0 :0
uid   : 26523480    : 10888902  : 1000001 : 3464494 0 : 0 : 3583 : 0 : 0 : 0


          --- Operation Processing Time ---

Op Type   : Total Ops : Avg Resp Time (ms)
----------:-----------:-------------------
Add       : 0         : 0.0
Bind      : 0         : 0.0
Compare   : 0         : 0.0
Delete    : 0         : 0.0
Modify    : 0         : 0.0
Modify DN : 0         : 0.0
Search    : 3583      : 117.58
All       : 3583      : 117.58


          --- Work Queue ---

           : Recent : Average : Maximum
-----------:--------:---------:--------
Queue Size : 0      : 0       : 1
% Busy     : 0      : 1       : 19
```

About the Monitor Entries

While the Directory Proxy Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the `cn=monitor` subtree. The types of monitor entries that are available include:

- **General Monitor Entry (cn=monitor)** – Provides a basic set of general information about the server.
- **Active Operations Monitor Entry (cn=Active Operations,cn=monitor)** – Provides information about all operations currently in progress in the server.

- **Backend Monitor Entries (cn={id} Backend,cn=monitor)** – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- **Client Connections Monitor Entry (cn=Client Connections,cn=monitor)** – Provides information about all connections currently established to the server.
- **Connection Handler Monitor Entry (cn={name},cn=monitor)** – Provides information about the configuration of each connection handler and the client connections established to it.
- **Database Environment Monitor Entries (cn={id} Database Environment,cn=monitor)** – Provides statistics and other data from the Oracle Berkeley DB Java Edition database envi- ronment used by the associated backend.
- **Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor)** – Provides infor- mation about the amount of usable disk space available to server components.
- **JVM Memory Usage Monitor Entry (cn=JVM Memory Usage,cn=monitor)** – Provides information about garbage collection activity, the amount of memory available to the server, and the amount of memory consumed by various server components.
- **JVM Stack Trace Monitor Entry (cn=JVM Stack Trace,cn=monitor)** – Provides a stack trace of all threads in the JVM.
- **LDAP Statistics Monitor Entries (cn={name} Statistics,cn=monitor)** – Provides information about the number of each type of operation requested and bytes transferred over the connection handler.
- **Processing Time Histogram Monitor Entry (cn=Processing Time Histogram,cn=monitor)** – Provides information about the number of percent of operations that completed in various response time categories.
- **SSL Context Monitor Entry (cn=SSL Context,cn=monitor)** – Provides information about the available and supported SSL Cipher Suites and Protocols on the server.
- **System Information Monitor Entry (cn=System Information,cn=monitor)** – Provides information about the underlying JVM and system.
- **Version Monitor Entry (cn=Version,cn=monitor)** – Provides information about the Directory Proxy Server version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Directory Proxy Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Using the Monitoring Interfaces

The PingDirectoryProxy Server exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the Administrative Console, JMX, over LDAP, using the LDAP SDK, and using SNMP.

Monitoring with the Administrative Console

Ping Identity has a Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Proxy Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.
4. Use the top level navigation dropdown and select 'Status.'
5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

The PingDirectoryProxy Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the Administrative Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The PingDirectoryProxy Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Proxy Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Proxy Server using the `dsconfig` tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

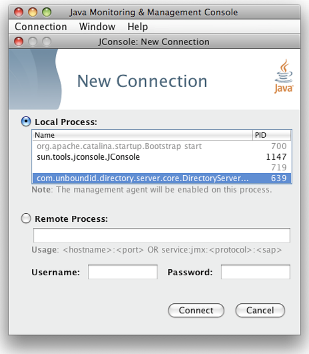
To Run JConsole

1. Use `JConsole` to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: `jconsole PID`. Or you can run JConsole remotely using: `jconsole hostname:port`.

   ```
   $ jconsole
   ```

   Note: If SSL is configured on the JMX Connection Handler, you must specify the Directory Proxy Server jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

   ```
   $ jconsole \
     -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
     -J-Djavax.net.ssl.trustStorePassword=secret \
     -J-Djava.class.path=$SERVER_ROOT/lib/PingDirectoryProxy.jar:/
   Library/Java/JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/
   jconsole.jar
   ```
2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.

   ● Local Process:

Name	PID
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1147
	719
com.unboundid.directory.server.core.DirectoryServer...	639

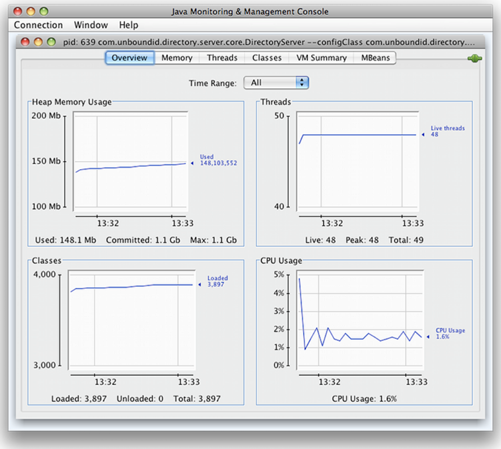
   Note: The management agent will be enabled on this process.

   ○ Remote Process:

   Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

   Username: Password:

3. Review the resource monitoring information.



Monitoring the Directory Proxy Server Using JConsole

You can set up JConsole to monitor the Directory Proxy Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscibe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Proxy Server using JConsole

1. Start the Directory Proxy Server.

```
$ bin/start-server
```

2. Enable the JMX Connection handler using the `dsconfig` tool. The handler is disabled by default. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

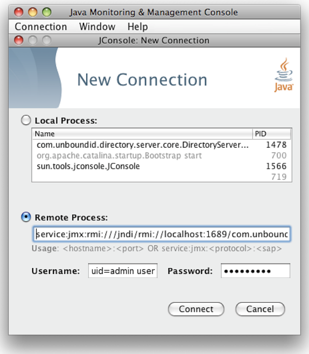
3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Proxy Server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



○ Local Process:

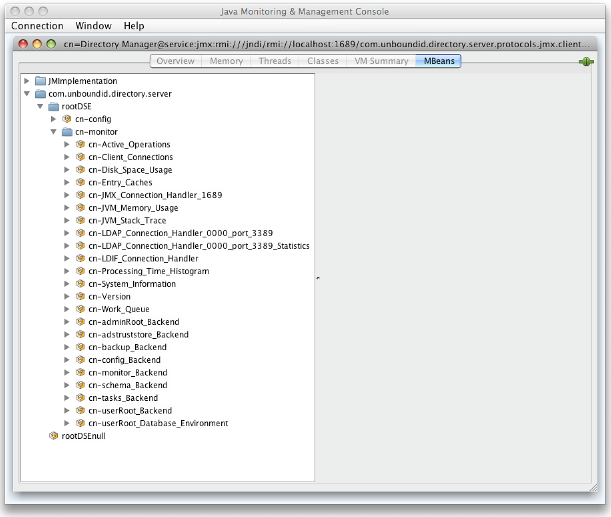
Name	PID
com.unboundid.directory.server.core.DirectoryServer...	1478
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1566
	719

● Remote Process: service:jmx:rmi:///jndi/rmi://localhost:1689/com.unbounc

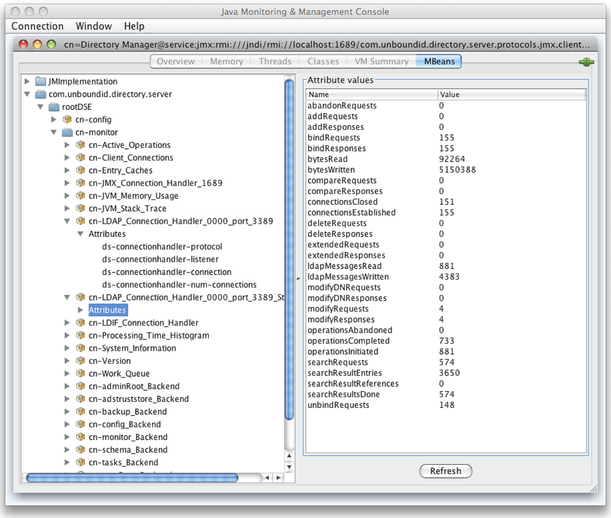
Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: uid=admin user Password: ••••••••••

6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.


Java Monitoring & Management Console
Connection Window Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview Memory Threads Classes VM Summary MBeans
JMImplementation
com.unboundid.directory.server
rootDSE
cn-config
cn-monitor
cn-Active_Operations
cn-Client_Connections
cn-Disk_Space_Usage
cn-Entry_Caches
cn-JMX_Connection_Handler_1689
cn-JVM_Memory_Usage
cn-JVM_Stack_Trace
cn-LDAP_Connection_Handler_0000_port_3389
cn-LDAP_Connection_Handler_0000_port_3389_Statistics
cn-LDIF_Connection_Handler
cn-Processing_Time_Histogram
cn-System_Information
cn-Version
cn-Work_Queue
cn-adminRoot_Backend
cn-adstruststore_Backend
cn-backup_Backend
cn-config_Backend
cn-monitor_Backend
cn-schema_Backend
cn-tasks_Backend
cn-userRoot_Backend
cn-userRoot_Database_Environment
rootDSEnull

7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.


Java Monitoring & Management Console
Connection Window Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview Memory Threads Classes VM Summary MBeans
JMImplementation
com.unboundid.directory.server
rootDSE
cn-config
cn-monitor
cn-Active_Operations
cn-Client_Connections
cn-Entry_Caches
cn-JMX_Connection_Handler_1689
cn-JVM_Memory_Usage
cn-JVM_Stack_Trace
cn-LDAP_Connection_Handler_0000_port_3389
Attributes
ds-connectionhandler-protocol
ds-connectionhandler-listener
ds-connectionhandler-connection
ds-connectionhandler-num-connections
cn-LDAP_Connection_Handler_0000_port_3389_St
Attributes
cn-LDIF_Connection_Handler
cn-Processing_Time_Histogram
cn-System_Information
cn-Version
cn-Work_Queue
cn-adminRoot_Backend
cn-adstruststore_Backend
cn-backup_Backend
cn-config_Backend
cn-monitor_Backend
cn-schema_Backend
cn-tasks_Backend
Refresh

Attribute values

Name	Value
abandonRequests	0
addRequests	0
addResponses	0
bindRequests	155
bindResponses	155
bytesRead	92264
bytesWritten	5150388
compareRequests	0
compareResponses	0
connectionsClosed	151
connectionsEstablished	155
deleteRequests	0
deleteResponses	0
extendedRequests	0
extendedResponses	0
ldapMessagesRead	881
ldapMessagesWritten	4383
modifyDNRequests	0
modifyDNResponses	0
modifyRequests	4
modifyResponses	4
operationsAbandoned	0
operationsCompleted	733
operationsInitiated	881
searchRequests	574
searchResultEntries	3650
searchResultReferences	0
searchResultsDone	574
unbindRequests	148

Monitoring over LDAP

The PingDirectoryProxy Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
 --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
 --baseDN "cn=monitor" "(objectclass=*)"
```

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Proxy Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " +
 String.valueOf(value));
      }
     }
     System.out.println();
  }
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring Using SNMP

The PingDirectoryProxy Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Proxy Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Proxy Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems, such as Alcatel-Lucent (Omnivista EMS), HP (OpenView), IBM-Tivoli (Netview), Oracle-Sun (Solstice Enterprise Manager), and others. Specific discussion on integrating an SNMP deployment on an NMS system is beyond the scope of this chapter. Consult with your NMS system for specific information.

The PingDirectoryProxy Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Proxy Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Proxy Server’s SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

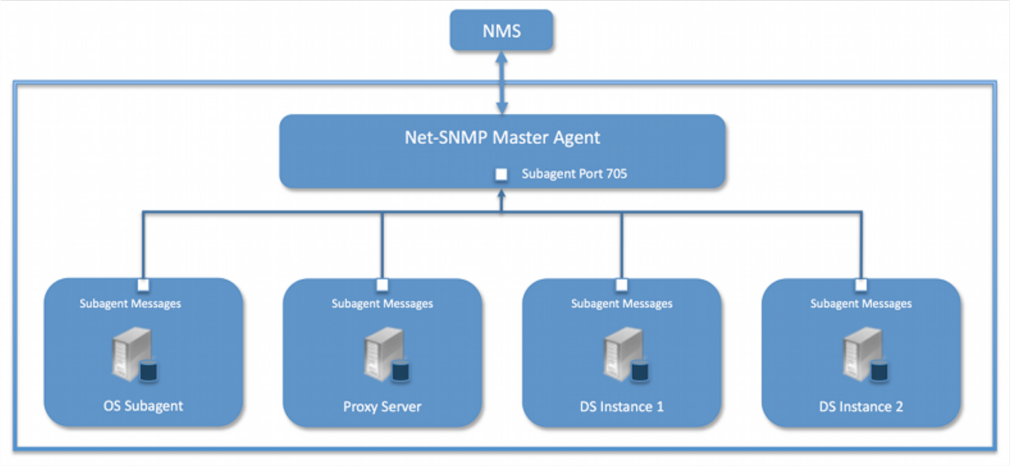


Figure 8: Example SNMP Deployment

One important note is that the PingDirectoryProxy Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note: The Directory Proxy Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Proxy Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Proxy Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" --set enabled:true
```

2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master
agent at localhost/705: Timeout
```

3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

```
master agentx agentXSocket tcp:localhost:705
```

Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

```
trap2sink localhost public 162
```

5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

```
rwuser initial
createUser initial MD5 setup_passphrase DES
```

6. Run the following command to create the SNMPv3 user.

```
snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
localhost create snmpuser initial
```

7. Start the `snmpd` daemon and after a few seconds you should see the following message in the Directory Proxy Server error log:

```
The SNMP subagent connected successfully to the master agent
at localhost:705. The SNMP context name is host.example.com:389
```

8. Set up a trap client to see the alerts that are generated by the Directory Proxy Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

```
authcommunity log, execute public
```

9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

10. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (-f), do not read any configuration files (-C) except the one specified with the -c option, print to standard output (-Lo), and then specify that debugging output should be turned on for the User-based Security Module (-Dusm). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
  -m all -M +/usr/share/snmp/mibs
```

11. Run the Net-SNMP client tools to test the feature. The following options are required: -v <SNMP version>, -u <user name>, -A <user password>, -l <security level>, -n <context name (instance name)> . The -m all option loads all MIBs in the default MIB directory in /usr/share/snmp/mibs so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389
 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n
 host.example.com:389 \
-m all localhost systemStatus
```

MIBS

The Directory Proxy Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Proxy Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Proxy Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Profiling Server Performance Using the Stats Logger

The Directory Proxy Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Directory Proxy Server ships with the built-in "Stats Logger' disabled. To enable it using the `dsconfig` tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select .

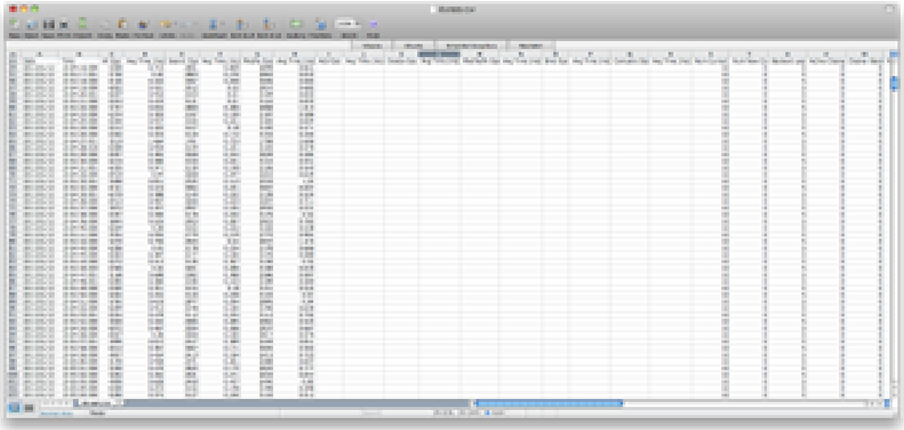
1. Run `dsconfig` in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plug-in.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

   ```
   >>>> Configure the properties of the Stats Logger Plugin

   Property                 Value(s)
   ```

```
------------------------------------------------------------------------
1)   description        Logs performance stats to a log file
                        periodically.
2)   enabled                     false
3)   local-db-backend-info       basic
4)   replication-info            basic
5)   entry-cache-info            -
6)   host-info                   -
7)   included-ldap-application   If per-application LDAP stats is enabled,
                                 then stats will be included for all
                                 applications.
8)   log-interval                1 s
9)   collection-interval         200 ms
10)  suppress-if-idle            true
11)  header-prefix-per-column    false
12)  empty-instead-of-zero       true
13)  lines-between-header        50
14)  included-ldap-stat          active-operations, num-connections,
                                 op-count-and-latency, work-queue
15)  included-resource-stat      memory-utilization
16)  histogram-format            count
17)  histogram-op-type           all
18)  per-application-ldap-stats  aggregate-only
19)  ldap-changelog-info         -
20)  gauge-info                  none
21)  log-file                    logs/dsstats.csv
22)  log-file-permissions        640
23)  append                      true
24)  rotation-policy             Fixed Time Rotation Policy, Size Limit
                                 Rotation Policy
25)  retention-policy            File Count Retention Policy

?)   help
f)   finish - apply any changes to the Periodic Stats Logger Plugin
a)   hide advanced properties of the Periodic Stats Logger Plugin
d)   display the equivalent dsconfig command lines to either re-create this
           object or only to apply pending changes
b)   back
q)   quit

Enter choice [b]:
```

7. Run the Directory Proxy Server. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate` --help to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in To Enable the Stats Logger.
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
```

```
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

Content-Reference to:

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```

2. On the Directory Proxy Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

    ```
    >>>> Configure the properties of the Custom Logged Stats
     >>>> via creating 'Memory Usage' Custom Logged Stats

             Property                   Value(s)
             ---------------------------------------------------------------
        1)   description                -
        2)   enabled                    true
        3)   monitor-objectclass        ds-memory-usage-monitor-entry
        4)   include-filter             -
        5)   attribute-to-log           total-bytes-used-by-memory-consumers
        6)   column-name                Memory Consumer Total (GB)
        7)   statistic-type             raw
        8)   header-prefix              -
        9)   header-prefix-attribute    -
        10)  regex-pattern              -
        11)  regex-replacement          -
        12)  divide-value-by            1073741824
        13)  divide-value-by-attribute  -
        14)  decimal-format             #.##
    ```

```
    15)  non-zero-implies-not-idle  false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

Content-Reference to:

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

Working with Alarms, Alerts, and Gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server unavailable' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and Administrative Console. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The Directory Proxy Server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

The Directory Proxy Server is compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when the Condition and Resource properties are the same. The Condition corresponds to the Summary column in the `admin-alerts-list.csv` file.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms.

To Test Alarms and Alerts

1. Configure a gauge with `dsconfig` and set the `override-severity` property to critical. The following example uses the CPU Usage (Percent) gauge.

```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

2. Run the `status` tool to verify that an alarm was generated with corresponding alerts. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status

                        --- Administrative Alerts ---
 Severity : Time            : Message
 ---------:-----------------:-------------------------------------------------------
 Info     : 11/Aug/2014     : A configuration change has been made in the
          : 15:48:46 -0500  : Directory Server:
          :                 : [11/Aug/2014:15:48:46.054 -0500]
          :                 : conn=17 op=73 dn='cn=Directory Manager,cn=Root
          :                 : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                 : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                 :  --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                 : --set warning-value:-1'
 Info     : 11/Aug/2014     : A configuration change has been made in the
          :  15:47:32 -0500 : Directory Server: [11/Aug/2014:15:47:32.547
 -0500]
          :                 : conn=4 op=196 dn='cn=Directory Manager,cn=Root
          :                 : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                 : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                 : --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                 :  --set warning-value:0'
 Error    : 11/Aug/2014     : Alarm [CPU Usage (Percent). Gauge CPU Usage
 (Percent)
          :  15:41:00 -0500 : for Host System has
          :                 : a current value of '18.583333333333332'.
          :                 : The severity is currently OVERRIDDEN in the
          :                 : Gauge's configuration to 'CRITICAL'.
          :                 : The actual severity is: The severity is
          :                 : currently 'NORMAL', having assumed this
 severity
          :                 : Mon Aug 11 15:41:00 CDT 2014. If CPU use is
 high,
          :                 : check the server's current workload and make
 any
          :                 : needed adjustments. Reducing the load on the
 system
          :                 : will lead to better response times.
          :                 : Resource='Host System']
          :                 : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
 hours
 Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
                         --- Alarms ---
 Severity : Severity Start : Condition : Resource    : Details
          : Time           :           :             :
 ---------:----------------:-----------:-------------:-------------------------------
 Critical : 11/Aug/2014    : CPU Usage : Host System : Gauge CPU Usage
 (Percent) for
          : 15:41:00 -0500 : (Percent) :             : Host System
          :                :           :             : has a current value
 of
          :                :           :             : '18.785714285714285'.
          :                :           :             : The severity is
 currently
          :                :           :             : 'CRITICAL', having
 assumed
          :                :           :             : this severity Mon Aug
 11
          :                :           :             : 15:49:00 CDT 2014. If
 CPU use
          :                :           :             : is high, check the
 server's
          :                :           :             : current workload and
 make any
          :                :           :             : needed adjustments.
 Reducing
          :                :           :             : the load on the
 system will
          :                :           :             : lead to better
 response times
 Warning  : 11/Aug/2014    : Work Queue: Work Queue  : Gauge Work Queue Size
 (Number
          : 15:39:40 -0500 : Size      :             : of Requests) for Work
 Queue
          :                : (Number of:             : has a current value
 of '27'.
          :                : Requests) :             : The severity is
 currently
          :                :           :             : 'WARNING' having
 assumed this
          :                :           :             : severity Mon Aug 11
 15:48:50
          :                :           :             : CDT 2014. If all
 worker
          :                :           :             : threads are busy
 processing
          :                :           :             : other client
 requests, then
          :                :           :             : new requests that
 arrive will
          :                :           :             : be forced to wait in
 the work
          :                :           :             : queue until a worker
 thread
```

```
          :                :           :             : becomes available
Shown are alarms of severity [Warning,Minor,Major,Critical]
Use the --alarmSeverity option to filter this list
```

Indeterminate Alarms

Indeterminate alarms are raised for a server condition for which a severity cannot be determined. In most cases these alarms are benign and do not issue alerts nor appear in the output of the `status` tool or Administrative Console by default. These alarms are usually caused by an enabled gauge that is intended to measure an aspect of the server that is not currently enabled. For example, gauges intended to monitor metrics related to replication may produce indeterminate alarms if a Directory Server is not currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There may be messages that can help determine the issue. The following is sample output from the `status` tool run with the –`alarmSeverity=indeterminate` option:

```
                        --- Alarms ---
Severity     : Severity Start : Condition      : Resource   : Details
             : Time           :                :            :
-------------:----------------:----------------:------------:------------------------
Normal       : 26/Aug/2014    : Startup Begun  : cn=config  : The Directory
 Server
             : 14:16:29 -0500 :                :            : is starting.
             :                :                :            :
Indeterminate: 26/Aug/2014    : Replication    : not        : The value of
 gauge
             : 14:16:40 -0500 : Latency        : available  : Replication
 Latency
             :                : (Milliseconds) :            : (Milliseconds)
 could not
             :                :                :            : be determined.
 The
             :                :                :            : severity is
 INDETERMINATE,
             :                :                :            : having assumed
 this
             :                :                :            : severity Tue Aug
 26
             :                :                :            : 14:17:10 CDT
 2014.
```

The following is an indeterminate alarm for the Replication Latency (Milliseconds) gauge. The following is a sample search of the monitor backend for this gauge’s entry. The result is an error message may explain the indeterminate severity:

```
# ldapsearch -w password --baseDN "cn=monitor"  \
-D"cn=directory manager" gauge-name="Replication Latency (Milliseconds)"

dn: cn=Gauge Replication Latency (Milliseconds),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-numeric-gauge-monitor-entry
objectClass: ds-gauge-monitor-entry
objectClass: extensibleObject
cn:           Gauge Replication Latency (Milliseconds)
gauge-name:  Replication Latency (Milliseconds)
resource:
severity:    indeterminate
summary:     The value of gauge Replication Latency (Milliseconds) could not
             be determined. The severity is INDETERMINATE, having assumed
             this severity Tue Aug 26 15:42:40 CDT 2014
error-message: No entries were found under cn=monitor having object
```

```
               class ds-replica-monitor-entry
              …
```

Working with Administrative Alert Handlers

The PingDirectoryProxy Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Proxy Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler**. Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Directory Proxy Server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler**. Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. Ping Identity uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler**. Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler**. Provides alert handler implementations created in third-party code using the Server SDK.

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the `dsconfig` tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using `dsconfig` in interactive command-line mode, which is visible on the "Standard" object menu.

To Configure the JMX Connection Handler

1. Use `dsconfig` to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the `ldapmodify` tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
givenName: JMX
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
```

```
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

To Configure the JMX Alert Handler

- Use dsconfig to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the dsconfig tool.

Configuring the SMTP Alert Handler

- Use the dsconfig tool to configure the SMTP Alert Handler.

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the dsconfig tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP2c. For more information on SNMP, see Monitoring Using SNMP.

To Configure the SNMP Subagent Alert Handler

- Use dsconfig to configure the SNMP subagent alert handler. The server-host-name is the address of the system running the SNMP subagent. The server-port is the port number on which the subagent is running. The community-name is the name of the SNMP community that is used for the traps.

  The Directory Proxy Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

Working with Virtual Attributes

The PingDirectoryProxy Server provides dynamically generated attributes called virtual attributes for local Directory Proxy Server data. The proxy virtual attributes apply to a local proxy backend, such as cn=config or the Root DSE. If you want to have virtual attributes in entries for proxied requests, then they must be configured in the backend servers. Alternately, attributes may be inserted into those entries using proxy transformations. For more information about configuring proxy transformations, see “Configuring Proxy Transformations”.

For example, you can define a virtual attribute and assign it to the Root DSE as follows:

```
$ bin/dsconfig create-virtual-attribute \
  --name defineDescriptionOnRootDSE --type user-defined \
  --set enabled:true --set attribute-type:description \
  --set filter:objectclass=ds-root-dse --set value:PrimaryProxy
```

If you search the Root DSE using the following LDAP search, you see that the description attribute now has the value `PrimaryProxy`.

```
$ bin/ldapsearch --baseDN "" --searchScope base --bindDN "" \
  --bindPassword "" --port 5389 -- hostname localhost \
  "objectclass=*" description

dn:
description:PrimaryProxy
```

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Proxy Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Chapter 9

Managing Monitoring

Topics:

The PingDirectoryProxy Server also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces via the PingData® Data Metrics Server, the Administrative Console, SNMP, JMX, and over LDAP. The Directory Proxy Server also provides a tool, the Periodic Stats Logger, to profile server performance.

This chapter presents the following information:

The Monitor Backend

The Directory Proxy Server exposes its monitoring information under the `cn=monitor` entry. Administrators can use various means to monitor the servers, including the PingData Data Metrics Server, through SNMP, the Administrative Console, JConsole, LDAP command-line tools, and the Periodic Stats Logger. Use the `bin/status` tool to display server component activity and state.

The list of all monitor entries can be seen using ldapsearch as follows:

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
 --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
 --baseDN "cn=monitor" "(objectclass=*)" cn
```

The following table describes a subset of the monitor entries:

Table 11: Directory Proxy Server Monitoring Components

Component	Description
Active Operations	Provides information about the operations currently being processed by the Directory Proxy Server. Shows the number of operations, information on each operation, and the number of active persistent searches.
Backends	Provides general information about the state of an a Directory Proxy Server backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size.
Client Connections	Provides information about all client connections to the Directory Proxy Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., connID="15", connectTime="20100308223038Z", etc.)
Connection Handlers	Provides information about the available connection handlers on the Directory Proxy Server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
Disk Space Usage	Provides information about the disk space available to various components of the Directory Proxy Server.
General	Provides general information about the state of the Directory Proxy Server, including product name, vendor name, server version, etc.
Index	Provides on each index. The monitor captures the number of keys preloaded, and counters for read/write/remove/open-cursor/read-for-search. These counters provide insight into how useful an index is for a given workload.
HTTP/HTTPS Connection Handler Statistics	Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc.

Component	Description
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc.
System Information	Provides general information about the system and the JVM on which the Directory Proxy Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Directory Proxy Server version, including build ID, version, revision number, etc.
Work Queue	Provides information about the state of the Directory Proxy Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, and number of busy worker threads. The work queue configuration has a `monitor-queue-time` property set to `true` by default. This logs messages for new operations with a `qtime` attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

Monitoring Disk Space Usage

The disk space usage monitor provides information about the amount of usable disk space available for Directory Proxy Server components. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface by various components of the server. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `/config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the DiskSpaceConsumer interface.

The disk space usage monitor provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

Three thresholds can be configured for this monitor:

- **Low space warning threshold**. This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Directory Proxy Server will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The Directory Proxy Server will also generate additional alerts as the amount of usable space is further reduced (e.g., each time the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.
- **Low space error threshold**. This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. Once the server enters this mode, then an administrator will have to

take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server will resume normal operation. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.

- **Out of space error threshold**. This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the PingDirectoryProxy Server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.
- **Disk space monitoring for tools**. The server monitors disk space consumption during processing for the `export-ldif`, `rebuild-index`, and `backup` tools. Space is monitored every 10 seconds if usable space for all monitored paths is greater than 15 percent of the capacity of those volumes. If usable space for any path drops below 15 percent, or below 10GB free, the space check frequency is increased to every second. Warning messages are generated if available space falls below 10 percent, or below 5GB free. If usable space for any path drops below two percent, or 1GB free, the tool processing is aborted and files may be removed to free up space.

The default configuration uses the same values for the low space error threshold and out of space error threshold. This is to prevent having the server online but rejecting requests, which will cause problems with applications trying to interact with the server. The low space warning threshold generates an alert before the problem becomes serious, well in advance of available disk space dropping to a point that it is critical.

The default values may not be suitable for all disk sizes, and should be adjusted to fit the deployment. Determining the best values should factor in the size of the disk, how big the database may become, how much space log files may consume, and how many backups will be stored.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring with the Data Metrics Server

The Data Metrics Server is an invaluable tool for collecting, aggregating and exposing historical and instantaneous data from the various Ping Identity servers in a deployment. The Data Metrics Server relies on a captive PostgreSQL data store for the metrics, which it collects from internal instrumentation across the instances, replicas, and data centers in your environment. The data is available via a Monitoring API that can be used to build custom dashboards and monitoring applications to monitor the overall health of your Ping Identity Platform system. For more information, see the *PingData Data Metrics Server Administration Guide*.

Monitoring Key Performance Indicators by Application

The PingDirectoryProxy Server can be configured to track many key performance metrics (for example, throughput and response-time) by the client applications requesting them. This feature is invaluable for measuring whether the Ping Identity identify infrastructure meets all of your service-level agreements (SLA) that have been defined for client applications.

When enabled, the per-application monitoring data can be accessed in the `cn=monitor` backend, the Periodic Stats Logger, and made available for collection by the PingData Data Metrics Server. See the “Profiling Server Performance Using the Periodic Stats Logger” for more information on using that component. Also, see the Directory Proxy Server Configuration section of the *PingData Data Metrics Server Administration Guide* for details on configuring the server to expose metrics that interest you. Tracked application information is exposed in the Data Metrics Server by metrics having the 'application-name' dimension. See the documentation under `docs/metrics` of the Data Metrics Server for information on which metrics are available with the 'application-name' dimension.

Configuring the External Servers

Before you install the Data Metrics Server, you need to configure the servers you will be monitoring: PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. The Data Metrics Server requires all servers to be version 3.5.0 or later. See the administration guides for each product for installation instructions.

Once you have installed the Directory Proxy Server, you can use the `dsconfig` tool to make configuration changes for the Data Metrics Server. When using the `dsconfig` tool interactively, set the complexity level to Advanced, so that you can make all the necessary configuration changes.

Preparing the Servers Monitored by the Data Metrics Server

The Metrics Backend manages the storage of metrics and provides access to the stored blocks of metrics via LDAP. The Metrics Backend is configured to keep a maximum amount of metric history based on log retention policies. The default retention policy uses the Default Size Limit Retention Policy, Free Disk Space Retention Policy, and the File Growth Limit Policy, limiting the total disk space used to 500 MB. This amount of disk typically contains more than 24 hours of metric history, which is ample. The Directory Proxy Server keeps a metric history so that the Data Metrics Server can be down for a period and then catch up when it comes back online.

The following two commands create a Retention Policy that limits the number of files to 2000, and sets the Metrics Backend to flush data to a new file every 30 seconds.

```
$ bin/dsconfig create-log-retention-policy \
  --policy-name StatsCollectorRetentionPolicy \
  --type file-count --set number-of-files:2000

$ bin/dsconfig set-backend-prop \
  --backend-name metrics --set sample-flush-interval:30s \
  --set retention-policy:StatsCollectorRetentionPolicy
```

These commands configure the Metrics Backend to keep 16 hours of metric history, which consumes about 250 MB of disk, ensuring that captured metrics are available to the Data Metrics Server within 30 seconds of when the metric was captured. The value of the `sample-flush-interval` attribute determines the maximum delay between when a metric is captured and when it can be picked up by the Data Metrics Server.

The flush interval can be set between 15 seconds and 60 seconds, with longer values resulting in less processing load on the Data Metrics Server. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in the Dashboard Application. If you change the `sample-flush-interval` attribute to 60 seconds in the example above, then the Directory Proxy Server keeps 2000 minutes of history. Because the number of metrics produced per unit of time can vary depending on the configuration, no exact formula can be used to compute how much storage is required for each hour of history. However, 20 MB per hour is a good estimate.

Configuring the Processing Time Histogram Plugin

The Processing Time Histogram plugin is configured on each Directory Proxy Server and Directory Proxy Server as a set of histogram bucket ranges. When the bucket ranges for a histogram change, the Data Metrics Server notices the change and marks samples differently. This process allows for histograms with the same set of bucket definitions to be properly aggregated and understood when returned in a query. If different servers have different bucket definitions, then a single metric query cannot return histogram data from the servers.

You should try to keep the Processing Time Histogram bucket definitions the same on all servers. Having different definitions restricts the ability of the Data Metrics Server API to aggregate histogram data across servers and makes the results of a query asking "What percentage of the search requests took less than 12 milliseconds?" harder to understand.

For each server in your topology, you must set the separate-monitor-entry-per-tracked-application property of the processing time histogram plugin to true. This property must be set to expose per-application monitoring information under `cn=monitor`. When the separate-monitor-entry-per-tracked-application property is set to true, then the per-application-ldap-stats property must be set to per-application-only on the Stats Collector Plugin and vice versa.

For example, the following `dsconfig` command line sets the required properties of the Processing Time Histogram plugin:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Processing Time Histogram” \
  --set separate-monitor-entry-per-tracked-application:true
```

The following `dsconfig` command line sets the per-application-ldap-stats property of the Stats Collector plugin to per-application-only:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Stats Collector” \
 --set per-application-ldap-stats:per-application-only
```

Setting the Connection Criteria to Collect SLA Statistics by Application

If you want to collect data about your SLAs, you need to configure connection criteria for each Service Level Agreement that you want to track. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used when the server needs to perform matching based on connection-level properties, such as filtered logging. For assistance using connection criteria, contact your authorized support provider.

For example, imagine that we are interested in collecting statistics on data that is accessed by clients authenticating as the Directory Manager. We need to create connection criteria on the Directory Proxy Server that identifies any user authenticating as the Directory Manager. The connection criteria name corresponds to the application-name dimension value that clients will specify when accessing the data via the API. When you define the Connection Criteria, change the included-user-base-dn property to include the Directory Manager’s full LDIF entry.

The following `dsconfig` command line creates connection criteria for the Directory Manager:

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name “Directory Manager” \
  --type simple \
  --set “included-user-base-dn:cn=Directory Manager,cn=Root DNs,cn=config”
```

Updating the Global Configuration

You also need to create Global Configuration-tracked applications for each app (connection criteria) you intend to track. The `tracked-application` property allows individual applications to be identified in the server by connection criteria. The name of the tracked application is the same as the name you defined for the connection criteria.

For example, the following `dsconfig` command line adds the connection criteria we created in the previous step to the list of tracked applications:

```
$ bin/dsconfig set-global-configuration-prop \
  --set "tracked-application:Directory Manager”
```

The value of the `tracked-application` field corresponds to the value of the `application-name` dimension value that clients will specify when accessing the data via the API.

Proxy Considerations for Tracked Applications

In a proxy environment, the criteria should be defined in the Directory Proxy Server since the Directory Proxy Server passes the application name through to the Directory Server in the intermediate client control. If a client of the Directory Proxy Server or Directory Server happens to use the intermediate client control, then the client name specified in the control will be used as the application name regardless of the criteria listed in the tracked-application property.

Monitoring Using SNMP

The PingDirectoryProxy Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Proxy Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Proxy Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems, such as Alcatel-Lucent (Omnivista EMS), HP (OpenView), IBM-Tivoli (Netview), Oracle-Sun (Solstice Enterprise Manager), and others. Specific discussion on integrating an SNMP deployment on an NMS system is beyond the scope of this chapter. Consult with your NMS system for specific information.

The PingDirectoryProxy Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Proxy Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Proxy Server's SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

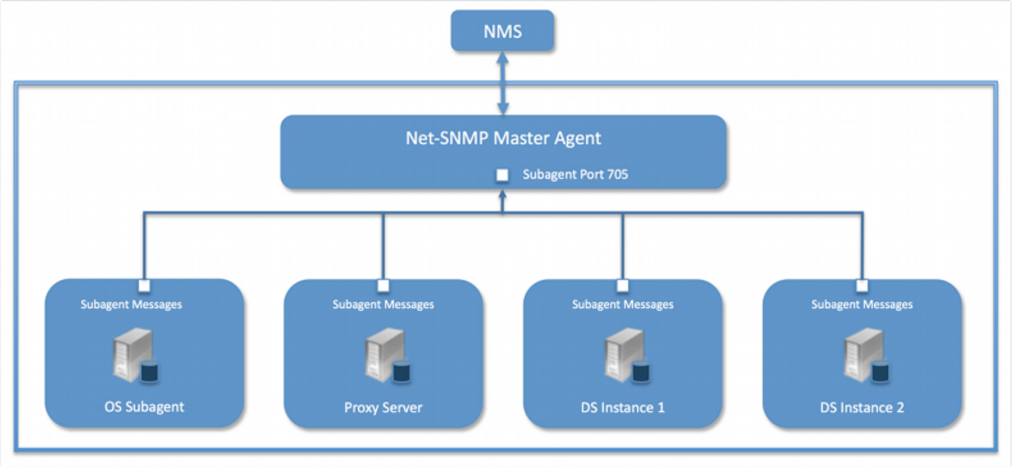


Figure 9: Example SNMP Deployment

One important note is that the PingDirectoryProxy Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default.

Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note: The Directory Proxy Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Proxy Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Proxy Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" --set enabled:true
```

2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master
agent at localhost/705: Timeout
```

3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

```
master agentx agentXSocket tcp:localhost:705
```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

```
trap2sink localhost public 162
```

5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

```
rwuser initial
createUser initial MD5 setup_passphrase DES
```

6. Run the following command to create the SNMPv3 user.

```
snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
localhost create snmpuser initial
```

7. Start the `snmpd` daemon and after a few seconds you should see the following message in the Directory Proxy Server error log:

```
The SNMP subagent connected successfully to the master agent
at localhost:705. The SNMP context name is host.example.com:389
```

8. Set up a trap client to see the alerts that are generated by the Directory Proxy Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

```
authcommunity log, execute public
```

9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

10. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (-f), do not read any configuration files (-C) except the one specified with the -c option, print to standard output (-Lo), and then specify that debugging output should be turned on for the User-based Security Module (-Dusm). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
  -m all -M +/usr/share/snmp/mibs
```

11. Run the Net-SNMP client tools to test the feature. The following options are required: -v <SNMP version>, -u <user name>, -A <user password>, -l <security level>, -n <context name (instance name)> . The -m all option loads all MIBs in the default MIB directory in /usr/share/snmp/mibs so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389
 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n
 host.example.com:389 \
-m all localhost systemStatus
```

MIBS

The Directory Proxy Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Proxy Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Proxy Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Monitoring with the Administrative Console

Ping Identity has a Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Proxy Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.
4. Use the top level navigation dropdown and select 'Status.'
5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

The PingDirectoryProxy Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the Administrative Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The PingDirectoryProxy Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Proxy Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Proxy Server using the `dsconfig` tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

To Run JConsole

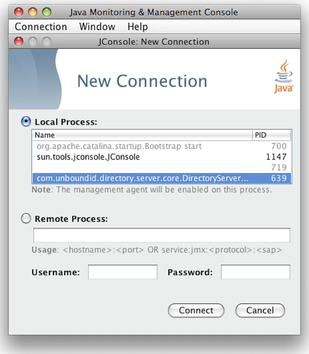
1. Use `JConsole` to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: `jconsole PID`. Or you can run JConsole remotely using: `jconsole hostname:port`.

   ```
   $ jconsole
   ```

   Note: If SSL is configured on the JMX Connection Handler, you must specify the Directory Proxy Server jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

   ```
   $ jconsole \
     -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
     -J-Djavax.net.ssl.trustStorePassword=secret \
     -J-Djava.class.path=$SERVER_ROOT/lib/PingDirectoryProxy.jar:/
   Library/Java/JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/
   jconsole.jar
   ```

2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.

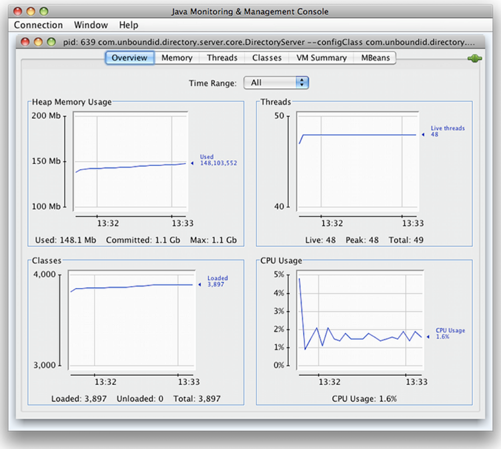


Name	PID
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1147
	719
com.unboundid.directory.server.core.DirectoryServer...	639

Note: The management agent will be enabled on this process.

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

3. Review the resource monitoring information.



Monitoring the Directory Proxy Server Using JConsole

You can set up JConsole to monitor the Directory Proxy Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscibe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Proxy Server using JConsole

1. Start the Directory Proxy Server.

```
$ bin/start-server
```

2. Enable the JMX Connection handler using the `dsconfig` tool. The handler is disabled by default. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

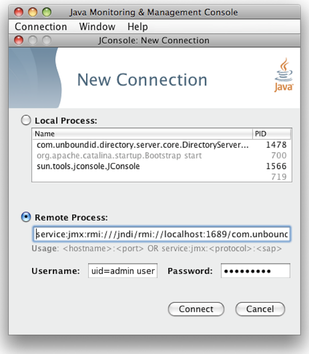
```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
```

```
ds-privilege-name: jmx-notify
```

4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Proxy Server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



○ Local Process:

Name	PID
com.unboundid.directory.server.core.DirectoryServer...	1478
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1566
	719

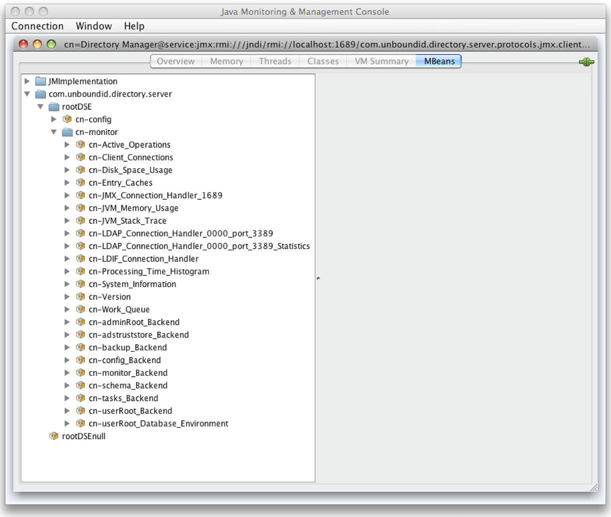
● Remote Process:

service:jmx:rmi:///jndi/rmi://localhost:1689/com.unbounc

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

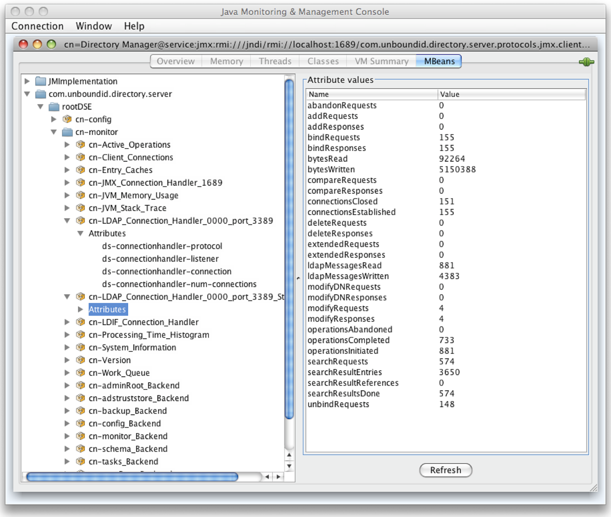
Username: uid=admin user Password: ••••••••••

6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.


Java Monitoring & Management Console
Connection
Window
Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview
Memory
Threads
Classes
VM Summary
MBeans

```
▶ JMImplementation
▼ com.unboundid.directory.server
  ▼ rootDSE
    ▶ cn-config
    ▼ cn-monitor
      ▶ cn-Active_Operations
      ▶ cn-Client_Connections
      ▶ cn-Disk_Space_Usage
      ▶ cn-Entry_Caches
      ▶ cn-JMX_Connection_Handler_1689
      ▶ cn-JVM_Memory_Usage
      ▶ cn-JVM_Stack_Trace
      ▶ cn-LDAP_Connection_Handler_0000_port_3389
      ▶ cn-LDAP_Connection_Handler_0000_port_3389_Statistics
      ▶ cn-LDIF_Connection_Handler
      ▶ cn-Processing_Time_Histogram
      ▶ cn-System_Information
      ▶ cn-Version
      ▶ cn-Work_Queue
      ▶ cn-adminRoot_Backend
      ▶ cn-adstruststore_Backend
      ▶ cn-backup_Backend
      ▶ cn-config_Backend
      ▶ cn-monitor_Backend
      ▶ cn-schema_Backend
      ▶ cn-tasks_Backend
      ▶ cn-userRoot_Backend
      ▶ cn-userRoot_Database_Environment
    rootDSEnull
```

7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.


Java Monitoring & Management Console
Connection
Window
Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview
Memory
Threads
Classes
VM Summary
MBeans
Refresh

```
JMImplementation
com.unboundid.directory.server
  rootDSE
    cn-config
    cn-monitor
      cn-Active_Operations
      cn-Client_Connections
      cn-Entry_Caches
      cn-JMX_Connection_Handler_1689
      cn-JVM_Memory_Usage
      cn-JVM_Stack_Trace
      cn-LDAP_Connection_Handler_0000_port_3389
        Attributes
          ds-connectionhandler-protocol
          ds-connectionhandler-listener
          ds-connectionhandler-connection
          ds-connectionhandler-num-connections
      cn-LDAP_Connection_Handler_0000_port_3389_St
        Attributes
      cn-LDIF_Connection_Handler
      cn-Processing_Time_Histogram
      cn-System_Information
      cn-Version
      cn-Work_Queue
      cn-adminRoot_Backend
      cn-adstruststore_Backend
      cn-backup_Backend
      cn-config_Backend
      cn-monitor_Backend
      cn-schema_Backend
      cn-tasks_Backend
```

Attribute values

Name	Value
abandonRequests	0
addRequests	0
addResponses	0
bindRequests	155
bindResponses	155
bytesRead	92264
bytesWritten	5150388
compareRequests	0
compareResponses	0
connectionsClosed	151
connectionsEstablished	155
deleteRequests	0
deleteResponses	0
extendedRequests	0
extendedResponses	0
ldapMessagesRead	881
ldapMessagesWritten	4383
modifyDNRequests	0
modifyDNResponses	0
modifyRequests	4
modifyResponses	4
operationsAbandoned	0
operationsCompleted	733
operationsInitiated	881
searchRequests	574
searchResultEntries	3650
searchResultReferences	0
searchResultsDone	574
unbindRequests	148

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Proxy Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " +
 String.valueOf(value));
      }
     }
     System.out.println();
  }
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring over LDAP

The PingDirectoryProxy Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Profiling Server Performance Using the Stats Logger

The Directory Proxy Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Directory Proxy Server ships with the built-in "Stats Logger' disabled. To enable it using the `dsconfig` tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select .

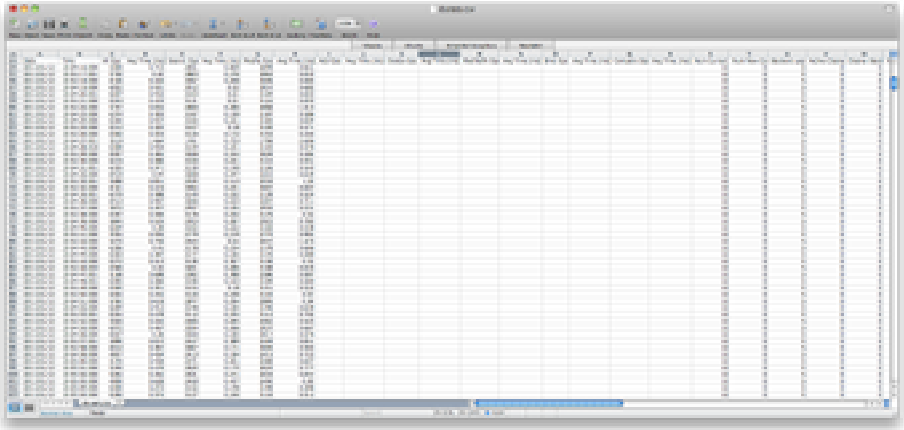
1. Run `dsconfig` in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plug-in.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

   ```
   >>>> Configure the properties of the Stats Logger Plugin

   Property                Value(s)
   ------------------------------------------------------------------------
   1)   description        Logs performance stats to a log file
                           periodically.
   2)   enabled                     false
   3)   local-db-backend-info       basic
   4)   replication-info            basic
   5)   entry-cache-info            -
   6)   host-info                   -
   7)   included-ldap-application   If per-application LDAP stats is enabled,
                                    then stats will be included for all
                                    applications.
   8)   log-interval                1 s
   9)   collection-interval         200 ms
   10)  suppress-if-idle            true
   11)  header-prefix-per-column    false
   12)  empty-instead-of-zero       true
   13)  lines-between-header        50
   14)  included-ldap-stat          active-operations, num-connections,
                                    op-count-and-latency, work-queue
   15)  included-resource-stat      memory-utilization
   16)  histogram-format            count
   17)  histogram-op-type           all
   18)  per-application-ldap-stats  aggregate-only
   19)  ldap-changelog-info         -
   20)  gauge-info                  none
   ```

```
21)   log-file                       logs/dsstats.csv
22)   log-file-permissions           640
23)   append                         true
24)   rotation-policy                Fixed Time Rotation Policy, Size Limit
                                     Rotation Policy
25)   retention-policy               File Count Retention Policy

?)    help
f)    finish - apply any changes to the Periodic Stats Logger Plugin
a)    hide advanced properties of the Periodic Stats Logger Plugin
d)    display the equivalent dsconfig command lines to either re-create this
            object or only to apply pending changes
b)    back
q)    quit

Enter choice [b]:
```

7. Run the Directory Proxy Server. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate` --help to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in To Enable the Stats Logger.
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

Content-Reference to:

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. On the Directory Proxy Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.

12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                    Value(s)
         ---------------------------------------------------------------
    1)   description                 -
    2)   enabled                     true
    3)   monitor-objectclass         ds-memory-usage-monitor-entry
    4)   include-filter              -
    5)   attribute-to-log            total-bytes-used-by-memory-consumers
    6)   column-name                 Memory Consumer Total (GB)
    7)   statistic-type              raw
    8)   header-prefix               -
    9)   header-prefix-attribute     -
    10)  regex-pattern               -
    11)  regex-replacement           -
    12)  divide-value-by             1073741824
    13)  divide-value-by-attribute   -
    14)  decimal-format              #.##
    15)  non-zero-implies-not-idle   false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

Content-Reference to:

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

Chapter

10

Troubleshooting the Directory Proxy Server

Topics:

This chapter provides the common problems and potential solutions that might occur when running PingDirectoryProxy Server. It is primarily targeted at cases in which the Directory Proxy Server is running on Linux® systems, but much of the information can be useful on other platforms as well.

This chapter presents the following information:

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the `jstat` utility with the -gc option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

To run the Directory Proxy Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`bin/start-server.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

Working with the Troubleshooting Tools

If problems arise with the Directory Proxy Server (whether from issues in the Directory Proxy Server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

Working with the Collect Support Data Tool

The Directory Proxy Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the `collect-support-data` tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool varies between systems. However, the tool always tries to get the same information across all systems for the target Directory Proxy Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Available Tool Options

The `collect-support-data` tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.
- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.
- **--sequential**. Use this option to diagnose “Out of Memory” errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.

- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, `vmstat`, `iostat`, or `mpstat`). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.
- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, `mpstat`). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.
- **--maxJstacks {number}**. Specifies the number of jstack samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they will impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. The comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain `dsreplication status` information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain `dsreplication status` information.

To Run the Collect Support Data Tool

1. Go to the server root directory.
2. Use the `collect-support-data` tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --serverRoot /opt/PingDirectoryProxy --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Directory Proxy Server Troubleshooting Tools

The PingDirectoryProxy Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server Version Information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Proxy Server software that is in use. If the server is running, then this information can be obtained from the "`cn=Version,cn=monitor`" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.
- Source revision number from which the server was built.
- Build information including build ID with time stamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Jetty, JZlib, SNMP4j (SNMP4J, Agent, Agentx), Groovy, LDAP SDK for Java, and the Server SDK.

LDIF Connection Handler

The Directory Proxy Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This Directory Proxy Server does not exist by default, but if it is created and an LDIF file is placed in it that contains one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

Embedded Profiler

If the Directory Proxy Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the `dsconfig` tool or the Administrative Console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The `profile-action` property for this configuration object can have one of the following values:

- **start** – Indicates that the embedded profiler should start capturing data in the background.
- **stop** – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a `logs/profile{timestamp}` file.
- **cancel** – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the `profiler-viewer` tool. This tool can operate in either a text-based mode, in which case it dumps a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

To Invoke the Profile Viewer in Text-based Mode

- Run the `profile-viewer` command and specify the captured log file using the --fileName option.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

To Invoke the Profile Viewer in GUI Mode

- Run the `profile-viewer` command and specify the captured log file using the --fileName option. To invoke GUI mode, add the option --useGUI.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Troubleshooting Resources for Java Applications

Because the PingDirectoryProxy Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Proxy Server. In many cases, obtaining the full benefit of these tools requires access to the Directory Proxy Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java Troubleshooting Tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the PingDirectoryProxy Server.

jps

The `jps` tool is a Java-specific version of the UNIX `ps` tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Proxy Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- **-v** – Includes the arguments passed to the JVM for the processes that are listed.
- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

jstack

The `jstack` tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can have user-friendly names (as do the threads used by the PingDirectoryProxy Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where {processID} is the process ID of the target JVM as returned by the `jps` command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the java command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The "-l" option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The “-m” option can be used to include native frames in the stack trace.

jmap

The `jmap` tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native `pmap` tool provided by many operating systems. As with the `jstack` tool, `jmap` can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the `jhat` tool

- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.
- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the ":live" portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

jhat

The `jhat` (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory Proxy Server encounters an out of memory error (as a result of the "-XX:+HeapDumpOnOutOfMemoryError" JVM option) or from the use of the `jmap` command with the "-dump" option.

The `jhat` tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The `jhat` tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the `jhat` web server to begin listening on port 7000. It can be accessed in a browser at `http://localhost:7000` (or `http://address:7000` from a remote system). An alternate port number can be specified using the "-port" option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL `http://localhost:7000/oql/` (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at `http://localhost:7000/oqlhelp/`.

jstat

The `jstat` tool is used to obtain a variety of statistical information from the JVM, much like the `vmstat` utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The {interval} option specifies the length of time in milliseconds between lines of output. The {processID} option specifies the process ID of the JVM used to run the Directory Proxy Server, which can be obtained by running `jps` as mentioned previously. The {type} option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.
- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.
- **-gccapacity** – Provides information about memory region capacities.

Java Diagnostic Information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the `jstat` utility with the -gc option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

To run the Directory Proxy Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`bin/start-server.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

JVM Crash Diagnostic Information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the base directory of the Directory Proxy Server installation. This file includes information on the underlying cause of the JVM crash, information about the threads running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system.

Troubleshooting Resources in the Operating System

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Proxy Server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the Directory Proxy Server, and in others system, tools can help narrow down the cause of the problem.

Identifying Problems with the Underlying System

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. To look for problems in the underlying system view the system log file (`/var/log/messages` on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.

Monitoring System Data Using the Data Metrics Server

The PingData Data Metrics Server provides collection and storage of performance data from your server topology. You can use the System Utilization Monitor with the Data Metrics Server to collect information about the host system CPU, disk, and network utilization on any platform except Linux. If you are not using the Data Metrics Server, you do not need to use the system utilization monitor. When data is being collected, it periodically forks the process and executes commands.

For more information about using the System Utilization Monitor, refer to the data collection chapter of the PingData Data Metrics Server documentation.

Examining CPU Utilization

Observing CPU utilization for the Directory Proxy Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU Utilization

To investigate CPU consumption of the system as a whole, use the `vmstat` command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Proxy Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Proxy Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU Utilization

To investigate CPU consumption on a per-CPU basis, use the `mpstat` command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "-P ALL" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but `mpstat` reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-Process Utilization

To investigate CPU consumption on a per-process basis, use a command such as the `top` utility on Linux. If a process other than the Java process used to run the Directory Proxy Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Proxy Server to run effectively.

Examining Disk Utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Proxy Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The `iostat` tool may be used to obtain information about the disk activity on the system.

On Linux systems, `iostat` should be invoked with the "-x" argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but to obtain an initial feel for how busy the underlying disks are, look at the "%util" column on Linux. This field shows the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Proxy Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a `tmpfs` filesystem).

Examining Process Details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard ps tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard ps command displays only a limited set of the arguments based on available space in the terminal window.

pstack

The pstack command can be used to obtain a native stack trace of all threads in a process. While a native stack trace might not be as user-friendly as a Java stack trace obtained using jstack, it includes threads that are not available in a Java stack trace. For example, the command displays those threads used to perform garbage collection and other housekeeping tasks. The general usage for the pstack command is:

```
pstack {processID}
```

dbx / gdb

A process debugger provides the ability to examine a process in detail. Like pstack, a debugger can obtain a stack trace for all threads in the process, but it also provides the ability to examine a process (or core file) in much greater detail, including observing the contents of memory at a specified address and the values of CPU registers in different frames of execution. The GNU debugger gdb is widely-used on Linux systems.

Note that using a debugger against a live process interrupts that process and suspends its execution until it detaches from the process. In addition, when running against a live process, a debugger has the ability to actually alter the contents of the memory associated with that process, which can have adverse effects. As a result, it is recommended that the use of a process debugger be restricted to core files and only used to examine live processes under the direction of your authorized support provider.

pfiles / lsof

To examine the set of files that a process is using (including special types of files, like sockets), you can use a tool such as lsof on Linux systems, (

```
lsof -p {processID}
```

)

Tracing Process Execution

Content-Reference to:

If a process is unresponsive but is consuming a nontrivial amount of CPU time, or if a process is consuming significantly more CPU time than is expected, it might be useful to examine the activity of that process in more detail than can be obtained using a point-in-time snapshot. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

This can be accomplished using the strace tool on Linux (

```
strace -f -p {processID}
```

).

Consult the strace manual page for additional information.

Problems with SSL Communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-server` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Examining Network Communication

Content-Reference to:

Because the PingDirectoryProxy Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Proxy Server itself can provide details about its interaction with clients by enabling debugging for the protocol or data debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the *tcpdump* tool on Linux, can be used to accomplish this.

There are many options that can be used with these tools, and their corresponding manual pages will provide a more thorough explanation of their use. However, to perform basic tracing to show the full details of the packets received for communication on port 389 with remote host 1.2.3.4, the following command can be used on Linux:

```

tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

It does not appear that the `tcpdump` tool provides support for LDAP parsing. However, it is possible to write capture data to a file rather than displaying information on the terminal (using "-w {path}" with `tcpdump`), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access.

Common Problems and Potential Solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General Methodology to Troubleshoot a Problem

When a problem is detected, Ping Identity recommends using the following general methodology to isolate the problem:

1. Run the `bin/status` tool or look at the server status in the Administrative Console. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts.
2. Look in the server logs. In particular, view the following logs:

   logs/errors
   logs/failed-ops
   logs/expensive-ops
3. Use system commands, such as `vmstat` and `iostat` to determine if the server is bottle-necked on a system resource like CPU or disk throughput.

4. For performance problem (especially intermittent ones like spikes in response time), enabling the `periodic-stats-logger` can help to isolate problems, because it stores important server performance information on a per-second basis. The `periodic-stats-logger` can save the information in a csv-formatted file that can be loaded into a spreadsheet. The information this logger makes available is very configurable. You can create multiple loggers for different types of information or a different frequency of logging (for example, hourly data in addition to per-second data). For more information, see "Profiling Server Performance Using the Periodic Stats Logger".
5. For replication problem, run `dsreplication status` and look at the `logs/replication` file.
6. For more advanced users, run the `collect-support-data` tool on the system, unzip the archive somewhere, and look through the collected information. This is often useful when administrators most familiar with the Ping Identity Platform do not have direct access to the systems where the production servers are running. They can examine the `collect-support-data` archive on a different server. For more information, see Using the Collect Support Data Tool.

Important: Run the `collect-support-data` tool whenever there is a problem whose cause is not easily identified, so that this information can be passed back to your authorized support provider before corrective action can be taken.

The Server Will Not Run Setup

If the `setup` tool does not run properly, some of the most common reasons include the following:

A Suitable Java Environment Is Not Available

The PingDirectoryProxy Server requires that Java be installed on the system and made available to the server, and it must be installed prior to running `setup`. If the `setup` tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the `setup` tool should be invoked with an explicitly-defined value for the *JAVA_HOME* environment variable that specifies the path to the Java installation that should be used. For example:

```
env JAVA_HOME=/ds/java ./setup
```

If this still does not work for some reason, then it can be that the value specified in the provided *JAVA_HOME* environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected Arguments Provided to the JVM

If the `setup` script attempts to launch the java command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running setup, but this might not be the case if either the *JAVA_ARGS* or *UNBOUNDID_JAVA_ARGS* environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run `setup`:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Used

The `setup` tool is only intended to provide the initial configuration for the Directory Proxy Server. It refuses to run if it detects that the `setup` tool has already been run, or if an attempt has been made to start the Directory Proxy Server prior to running the `setup` tool. This protects an existing Directory Proxy Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Proxy Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run `setup` in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run `setup` failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the `setup` program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The Server Will Not Start

If the Directory Proxy Server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running

Only a single instance of the Directory Proxy Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other administrative operations can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Proxy Server could not acquire an exclusive lock on file
/ds/PingDirectoryProxy/locks/server.lock: The exclusive lock requested for
 file
/ds/PingDirectoryProxy/locks/ server.lock was not granted, which indicates
that another process already holds a shared or exclusive lock on that
file. This generally means that another instance of this server is already
running
```

If the Directory Proxy Server is not running (and is not in the process of starting up or shutting down) and there are no other tools running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the `locks` directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the filesystem.

There Is Not Enough Memory Available

When the Directory Proxy Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Proxy Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Proxy Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Proxy Server might need to be re-configured to use a smaller amount of memory than had been previously configured.
- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Proxy Server, or reconfigure the Directory Proxy Server to reduce the amount of memory that it attempts to use.
- The Directory Proxy Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the `vmstat` command and wait until the amount of free memory stops growing before attempting to restart the server.
- If the system is configured with one or more memory-backed filesystems, then look to see if there are any large files that can be consuming a significant amount of memory in any of those locations. If so, then remove them or relocate them to a disk-based filesystem.

- For Linux systems only, if there is a mismatch between the huge pages setting for the JVM and the huge pages reserved in the operating system.

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An Invalid Java Environment or JVM Option Was Used

If an attempt to start the Directory Proxy Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the `config/java.properties` file to reference to path to the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.
- If an *UNBOUNDID_JAVA_HOME* or *UNBOUNDID_JAVA_BIN* environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the `config/java.properties` file. Similarly, if an *UNBOUNDID_JAVA_ARGS* environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the *UNBOUNDID_JAVA_HOME*, *UNBOUNDID_JAVA_BIN*, and *UNBOUNDID_JAVA_ARGS* environment variables before trying to start the server.

Note that any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, then it can be necessary to remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option Was Provided

There are a small number of arguments that are provided when running the `bin/start-server` command, but in most cases, none are required. If one or more command-line arguments were provided for the `bin/start-server` command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration

If a change is made to the Directory Proxy Server configuration using an officially-supported tool like `dsconfig` or the Administrative Console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Proxy Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the `logs/config-audit.log` file to see what recent configuration changes have been made with the server online, or in the `config/archived-configs` directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the --useLastKnownGoodConfig option:

```
$ bin/start-server --useLastKnownGoodConfig
```

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the `config/config.ldif` file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.

👉 **Note:** In addition to manually-editing the config file, you can look at previous achived configurations to see if the most recent one works. You can also use the `ldif-diff` tool to compare the configurations in the archive to the current configuration to see what is different.

You Do Not Have Sufficient Permissions

The Directory Proxy Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Proxy Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Proxy Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/PingDirectoryProxy
```

The Server Has Crashed or Shut Itself Down

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Proxy Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using “kill -9”), then messages are written to the `error` and `server.out` logs explaining the reason for the shutdown.
- The Directory Proxy Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the `uptime` command on the underlying system shows that it was recently booted.
- The Directory Proxy Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Proxy Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the `error` and `server.out` logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Proxy Server was running crashed. If this happens, then the JVM should dump a fatal error log (a `hs_err_pid{processID}.log` file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

Conditions for Automatic Server Shutdown

All PingData servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The Directory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
           --set unrecoverable-database-error-mode:initiate-server-shutdown
```

The Server Will Not Accept Client Connections

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Proxy Server is not running.
- The underlying system on which the Directory Proxy Server is installed is not running.
- The Directory Proxy Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.
- If the Directory Proxy Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.
- If the Directory Proxy Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Proxy Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The Server is Unresponsive

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the `jstack` command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

  A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

  👉 **Note:** If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.
- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is

able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the `jstack` command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.

- If the JVM in which the Directory Proxy Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a `pstack` of the native process should show that most threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.
- If the JVM in which the Directory Proxy Server is running has hung for some reason, then the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like `snoop` or `tcpdump` should show that packets sent to the system on which the Directory Proxy Server is running are not receiving TCP acknowledgement.
- If the system on which the Directory Proxy Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Proxy Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is Slow to Respond to Client Requests

If the Directory Proxy Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Proxy Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see "Profiling Server Performance Using the Periodic Stats Logger". The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second. You can also run the `bin/status` command or view the status in the Administrative Console to see the Directory Proxy Server’s Work Queue information (also see the next bullet point).
- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Proxy Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the `bin/status` command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

```
           --- Work Queue ---
           : Recent : Average : Maximum
-----------:--------:---------:--------
Queue Size : 10   : 1       : 10
% Busy     : 17     : 14      : 100
```

  You can also view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second.

- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, `iostat` reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see Disk-Bound Deployments.
- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.
- If the Directory Proxy Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, `iostat` should report very high utilization for disks used to hold swap space, and commands like `cat /proc/meminfo` on Linux can report a large amount of swap memory in use. Another cause of swapping is if swappiness is not set to 0 on Linux. For more information, see Disable File System Swapping.
- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Proxy Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server Returns Error Responses to Client Requests

If a large number of client requests are receiving error responses, then view the `logs/failed-ops` log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.
- If a portion of the Directory Proxy Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxies or load balancers or both to route requests away from the affected server. As of Directory Proxy Server version 3.1 or later, the Directory Proxy Server will indicate that it is in a degraded status and the Directory Proxy Server will route around it.
- If the Directory Proxy Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.
- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

The Server Must Disconnect a Client Connection

If a client connection must be disconnected due to the expense of the client's request, such as an unindexed search across a very large database, perform the following:

- Find the client's connection ID by looking in the `cn=Active Operations,cn=monitor monitor` entry.

```
$ bin/ldapsearch -baseDN cn=monitor "cn=active operations" \
  --bindDN "cn=directory manager"  \
  --bindPassword password
```

- The monitor entry will contain attribute values for `operation-in-progress`, which look like an access log message. Look for the value of `conn` in the client request that should be disconnected. In the following example, the client to be disconnected is requesting a search for `(description=expensive)`, which is on connection 6.

```
dn: cn=Active Operations,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-active-operations-monitor-entry
objectClass: extensibleObject
cn: Active Operations
num-operations-in-progress: 2
operation-in-progress: [15/Dec/2014:10:55:35 -0600] SEARCH conn=6 op=3
 msgID=4
     clientIP="10.8.4.21" authDN="cn=app1,ou=applications,dc=example,dc=com"
 base="dc
     =example,dc=com" scope=wholeSubtree filter="(description=expensive)"
 attrs="A
     LL" unindexed=true
operation-in-progress: [15/Dec/2014:10:56:11 -0600] SEARCH conn=7 op=1
 msgID=2
     clientIP="127.0.0.1" authDN="cn=Directory Manager,cn=Root
 DNs,cn=config" base="c
     n=monitor" scope=wholeSubtree filter="(cn=active operations)"
 attrs="ALL"
     num-persistent-searches-in-progress: 0
```

- With the connection ID value, create a file with the following contents, named `disconnect6.ldif`.

```
dn: ds-task-id=disconnect6,cn=scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-disconnect
ds-task-disconnect-connection-id: 6
ds-task-id: disconnect6
ds-task-class-name:
 com.unboundid.directory.server.tasks.DisconnectClientTask
```

- This LDIF file represents a task entry. The connection ID value 6 is assigned to `ds-task-disconnect-connection-id`. The value for `ds-task-id` value does not follow a specific convention. It must be unique among other task entries currently cached by the server.
- Disconnect the client and cancel the associated operation by adding the task entry to the server:

```
$ bin/ldapmodify --filename disconnect6.ldif  \
  --defaultAdd --bindDN "cn=directory manager" \
  --bindPassword password
```

Problems with the Administrative Console

If a problem arises when trying to use the Administrative Console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Proxy Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with the Directory Proxy Server instance using the Administrative Console, then the access and error logs for that Directory Proxy Server instance might provide additional information about the underlying problem.

Problems with the Administrative Console: JVM Memory Issues

Console runs out of memory (PermGen). An inadequate PermSize setting in the server, while hosting web applications like the Administrative Console may result in errors like this in the error log:

```
[02/Mar/2016:07:50:27.017 -0600] threadID=2 category=UTIL
    severity=SEVERE_ERROR msgID=-1 msg="The server experienced an unexpected
    error. Please report this problem and include this log file.
    OutOfMemoryError: PermGen space
    ()\ncom.unboundid.directory.server.core.DirectoryServer.uncaughtException
    (DirectoryServer.java:15783)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1057)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.Thread.dispatchUncaughtException
    (Thread.java:1986)\nBuild revision: 22496\n"
```

This is only relevant for servers running Java 7.

Global Index Growing Too Large

If the global index appears to be growing too large, you can reload from the backend directory servers. Use the `reload-index` tool with the --fromDS option, overriding the configuration of the `prime-index-source` property. You can do this on a one off basis if the global index appears to be growing too large as follows:

```
$ bin/reload-index \
  --bindPassword password \
  --baseDN "dc=example,dc=com" \
  --fromDS
```

Forgotten Proxy User Password

If you have forgotten the password you set for the `cn=Proxy User` entry, you can work around the problem as follows:

- You can temporarily add a second password to the proxy user entry so that you can transition all of the proxy server instances to the new password. However, you should have multiple passwords on the `cn=Proxy User` entry for the shortest time possible.
- If you do not know the clear-text value, then you can use the encrypted value when configuring the new Directory Proxy Server. The encryption scheme allows reversible passwords that are stored in the server configuration so that they can be decrypted by any server instance.
- You can create a new root user in the directory server instances with the appropriate set of privileges and have the new proxy server instance use that account to authenticate. Since it is not a good idea to have an account for which you do not know the password, you may want to update all of the other proxy server instances to use the new account.
- You can use a protocol analyzer like `snoop` or `Wireshark`, to capture the password from the network communication.

Providing Information for Support Cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the `collect-support-data` tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

Chapter

11

Managing the SCIM Servlet Extension

Topics:

The PingDirectoryProxy Server provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS.

This section describes fundamental SCIM concepts and provides information on configuring SCIM on your server.

Overview of SCIM Fundamentals

Understanding the basic concepts of SCIM can help you use the SCIM extension to meet the your deployment needs. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension implements the 1.1 version of the SCIM specification. Familiarize yourself with this specification to help you understand and make efficient use of the SCIM extension and the SCIM SDK. The SCIM specifications are located on the Simplecloud website.

Summary of SCIM Protocol Support

PingData implements all required features of the SCIM protocol and most optional features. The following table describes SCIM features and whether they are supported.

Table 12: SCIM Protocol Support

SCIM Feature	Supported
Etags	Yes
JSON	Yes
XML*	Yes
Authentication/Authorization	Yes, via HTTP basic authentication or OAuth 2.0 bearer tokens
Service Provider Configuration	Yes
Schema	Yes
User resources	Yes
Group resources	Yes
User-defined resources	Yes
Resource retrieval via GET	Yes
List/query resources	Yes
Query filtering*	Yes
Query result sorting*	Yes
Query result pagination*	Yes (Directory Server, not Directory Proxy Server)
Resource updates via PUT	Yes
Partial resource updates via PATCH*	Yes
Resource deletes via DELETE	Yes
Resource versioning*	Yes (requires configuration for updated servers)
Bulk*	Yes
HTTP method overloading	Yes
Raw LDAP Endpoints**	Yes

* denotes an optional feature of the SCIM protocol.

** denotes a PingData extension to the basic SCIM functionality.

About the Identity Access API

The PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server support an extension to the SCIM standard called the Identity Access API. The Identity Access API provides an alternative to LDAP by supporting CRUD (create, read, update, and delete) operations to access directory server data over an HTTP connection.

SCIM and the Identity Access API are provided as a unified service through the SCIM HTTP Servlet Extension. The SCIM HTTP Servlet Extension can be configured to only enable core SCIM resources (e.g., 'Users' and 'Groups'), only LDAP object classes (e.g., `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there may be two representations with different schemas for any resources defined in the `scim-resources.xml` file: the SCIM representation and the raw LDAP representation. Likewise, because resources are exposed by an LDAP object class, and because these are hierarchical (e.g., `top` --> `person` --> `organizationalPerson` --> `inetOrgPerson`, etc.), a client application can access an entry in multiple ways due to the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the PingDirectory Server.

Creating Your Own SCIM Application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM SDK is available for download at https://github.com/pingidentity/scim.

Note: The value of a read-only SCIM attribute can be set by a POST operation if the SCIM attribute is a custom attribute in the `scim-resource.xml` config file, but not if the SCIM attribute is a core SCIM attribute.

Configuring SCIM

This section discusses details about the PingData implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available on the Simplecloud website.

Before You Begin

To set up your SCIM servlet extension, the Directory Proxy Server provides a `dsconfig` batch file, `scim-config-ds.dsconfig`, that contains the necessary commands to set up a SCIM extension, which is located in the `config` directory. The batch file assumes that the JKS key and trust manager providers are already enabled. Setup examples are presented in later sections.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See Managing the SCIM Schema for more information.

Configuring the SCIM Servlet Extension

The Directory Proxy Server provides a default SCIM HTTP Servlet Extension that can be enabled and configured using a `dsconfig` batch script located in the `config` directory. The script runs a series of commands that enables an HTTPS Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension. There are additional optional configurations (e.g., changing the log format, enable `entryDN` virtual attribute and using VLV indexes) that you can make by altering the `dsconfig` batch script.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See Managing the SCIM Schema for more information.

The following procedures show how to configure SCIM on the server. The first example procedure shows the steps to manually configure SCIM without using the script. The second example procedure uses the `dsconfig` batch script to configure SCIM.

To Configure the SCIM Servlet Extension

1. Before you enable the SCIM servlet extension, add access controls on each of the backend Directory Servers to allow read access to operational attributes used by the SCIM Servlet Extension. We recommend using the following non-interactive command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id
 ||
    createTimestamp || modifyTimestamp")
    (version 3.0;acl "Authenticated read access to operational attributes \
    used by the SCIM servlet extension"; allow (read,search,compare)
    userdn="ldap:///all";)'
```

2. On the Directory Proxy Server, enable the SCIM servlet extension by running the `dsconfig` batch file.

```
$ bin/dsconfig --batch-file config/scim-config-proxy.dsconfig
```

3. The `dsconfig` batch file must be edited to use the correct request processor name and base DN name(s) for the `set-request-processor-prop` and `set-root-dse-backend-prop` commands, respectively, as described in the "Configuring LDAP Control Support on All Request Processors" and "SCIM Servlet Extension Authentication" sections later in the chapter.

To Enable Resource Versioning

Resource versioning is enabled by default in new installations. Upgraded servers that had SCIM enabled need additional configuration to enable resource versioning.

1. Enable the `ds-entry-checksum` virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
                        --name ds-entry-checksum \
                        --set enabled:true
```

2. Remove any existing access controls required by SCIM for read access to operational attributes:

```
$ bin/dsconfig set-access-control-handler-prop \
                        --remove 'global-aci:(targetattr="entryUUID ||
 entryDN || ds-entry-unique-id || createTimestamp || ds-create-time ||
 modifyTimestamp || ds-update-time")(version 3.0;acl "Authenticated read
 access to operational attributes used by the SCIM servlet extension"; allow
 (read,search,compare) userdn="ldap:///all"'
```

3. On the backend Directory Server, make sure new access controls required by SCIM for read access to operational attributes are enabled with the following command. If this ACI is not present, issues will occur when a SCIM client tries to authenticate with an non-root DN.

```
$ bin/dsconfig set-access-control-handler-prop \
                        --add  'global-aci:(targetattr="entryUUID ||
 entryDN || ds-entry-unique-id || createTimestamp || ds-create-time ||
 modifyTimestamp || ds-update-time || ds-entry-checksum")(version 3.0;acl
 "Authenticated read access to operational attributes used by the SCIM
 servlet extension"; allow (read,search,compare) userdn="ldap:///all"'
```

Configuring LDAP Control Support on All Request Processors (Proxy Only)

You need to configure support for the required LDAP controls on all request processors handling LDAP requests that result from SCIM requests. Change the request processor name that was provided as an example and repeat the command for all additional request processors.

To Configure LDAP Control Support on All Request Processors

- Use dsconfig to change the request processor name that was provided as an example and repeat the command for all additional request processors. Make sure to use your deployment's request processor name.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add supported-control-oid:1.2.840.113556.1.4.319 \
  --add supported-control-oid:1.2.840.113556.1.4.473 \
  --add supported-control-oid:2.16.840.1.113730.3.4.9
```

SCIM Servlet Extension Authentication

The SCIM servlet supports authentication using either the HTTP Basic authentication scheme, or OAuth 2.0 bearer tokens. When authenticating using HTTP Basic authentication, the SCIM servlet attempts to correlate the username component of the Authorization header to a DN in the Directory Proxy Server. If the username value cannot be parsed directly as a DN, it is correlated to a DN using an Identity Mapper. The DN is then used in a simple bind request to verify the password.

In deployments that use an OAuth authorization server, the SCIM extension can be configured to authenticate requests using OAuth bearer tokens. The SCIM extension supports authentication with OAuth 2.0 bearer tokens (per RFC 6750) using an OAuth Token Handler Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, PingData provides the token handler API to decode incoming bearer tokens and extract or correlate associated authorization DNs.

Neither HTTP Basic authentication nor OAuth 2.0 bearer token authentication are secure unless SSL is used to encrypt the HTTP traffic.

Enabling HTTPS Communications

If you want the SCIM HTTP connection handler to use SSL, which is mandated by the SCIM specification, you need to enable a Key Manager provider and Trust Manager provider.

To enable SSL during the Directory Proxy Server's initial setup, include the --ldapsPort and the --generateSelfSignedCertificate arguments with the `setup` command. If your server already has a certificate that you would like to use, set the `key-manager-provider` to the value you set when you enabled SSL in the Directory Proxy Server, or define a new key manager provider (see Configuring HTTP Connection Handlers).

To Configure Basic Authentication Using an Identity Mapper

By default, the SCIM servlet is configured to use the Exact Match Identity Mapper, which matches against the `uid` attribute. In this example, an alternate Identity Mapper is created so that clients can authenticate using `cn` values.

1. Create a new Identity Mapper that uses a match attribute of cn.

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "CN Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:cn
```

2. Configure the SCIM servlet to use the new Identity Mapper.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set "identity-mapper:CN Identity Mapper"
```

To Enable OAuth Authentication

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the Server SDK documentation.

1. Install your OAuth token handler on the server using `dsconfig`.

```
$ bin/dsconfig create-oauth-token-handler \
  --handler-name ExampleOAuthTokenHandler \
  --type third-party \
  --set extension-
class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
```

2. Configure the SCIM servlet extension to use it as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set oauth-token-handler:ExampleOAuthTokenHandler
```

Using HTTP Basic Authentication with Bare UID on the Directory Proxy Server

As discussed above, clients can authenticate to the SCIM extension using HTTP basic authentication and a bare UID value. However, when a SCIM extension is hosted by a Directory Proxy Server, the server needs to be explicitly configured with the names of subordinate base DNs to search. To do this, run the following command on the Directory Proxy Server for every base DN that may be accessed via SCIM. Make sure to specify your deployment's subordinate base DN.

```
$ bin/dsconfig set-root-dse-backend-prop \
  --set subordinate-base-dn:dc=example,dc=com
```

Verifying the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

To Verify the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

- Run `curl` to verify that the SCIM extension is running. The -k (or --insecure) option is used to turn off curl's verification of the server certificate, since the example Directory Proxy Server is using a self-signed certificate.

```
$ curl -u "cn=Directory Manager:password" \
-k "https://localhost:8443/scim/ServiceProviderConfigs"

{"schemas":["urn:scim:schemas:core:1.0"],"id":"urn:scim:schemas:core:1.0",
"patch":{"supported":true},"bulk":{"supported":true,"maxOperations":10000,
"maxPayloadSize":10485760},"filter":{"supported":true,"maxResults":100},
"changePassword":{"supported":true},"sort":{"supported":true},
"etag":{"supported":false},"authenticationSchemes":[{"name":"HttpBasic",
"description":"The HTTP Basic Access Authentication scheme. This scheme is
not considered to be a secure method of user authentication (unless used in
conjunction with some external secure system such as SSL), as the user
name and password are passed over the network as cleartext.","specUrl":
"http://www.ietf.org/rfc/rfc2617","documentationUrl":
"http://en.wikipedia.org/wiki/Basic_access_authentication"}]}
```

- If the user ID is a valid DN (such as `cn=Directory Manager`), the SCIM extension authenticates by binding to the Directory Proxy Server as that user. If the user ID is not a valid DN, the SCIM extension searches for an entry with that `uid` value, and binds to the server as that user. To verify authentication to the server as the user with the `uid` of `user.0`, run the following command:

```
$ curl -u "user.0:password" \
  -k "https://localhost:8443/scim/ServiceProviderConfigs"
```

Configuring Advanced SCIM Extension Features

The following sections show how to configure advanced SCIM servlet extension features, such as bulk operation implementation, mapping SCIM resource IDs, and transformations.

Managing the SCIM Schema

This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About SCIM Schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping LDAP Schema to SCIM Resource Schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Proxy Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

Note that LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). The easiest and most-correct way to handle this is to make sure that any attributes that may contain binary data are declared using "dataType=binary" in the `scim-resources.xml` file. Likewise, when using the Identity Access API make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes which may contain binary data. This will cause the server to automatically base64-encode the data before returning it to clients and will also make it predictable for clients because they can assume the data will always be base64-encoded.

However, it is still possible that attributes that are not declared as binary in the schema may contain binary data (or just data that is invalid in XML), and the server will always check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values which are found to include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the <resource> Element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A resource element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the <attribute> Element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as urn:scim:schemas:core:1.0. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is false.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:

    `simple`: specifies a simple, singular SCIM attribute.
    `complex`: specifies a complex, singular SCIM attribute.
    `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
    `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the <simple> Element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: binary, boolean, dateTime, decimal, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simple element contains the following XML element:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the <complex> Element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

About the <simpleMultivalued> Element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simpleMultiValued element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the <complexMultiValued> Element

A `complexMultiValued` element has the following XML attribute:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is address.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.
- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the <subAttribute> Element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: binary, boolean, dateTime, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the complexMultiValued element.

About the <canonicalValue> Element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the <mapping> Element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in the Mapping LDAP Entries to SCIM Using the SCIM-LDAP API section.

About the <subMapping> Element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. The available transformations are described in *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API*.

About the <LDAPSearch> Element

An `LDAPSearch` element contains the following XML elements in sequence:

- `baseDN`: a required element specifying one or more LDAP search base DNs to be used when querying for the SCIM resource.
- `filter`: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- `resourceIDMapping`: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN. Note The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

Note: The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

About the <resourceIDMapping> Element

The `resourceIDMapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `scim-consumer`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or Directory Proxy Server, meaning that a value is automatically provided by the Directory Proxy Server (as would be the case if the mapped LDAP attribute is entryUUID).

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the <LDAPAdd> Element

An LDAPAdd element contains the following XML elements in sequence:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

About the <fixedAttribute> Element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The value merge indicates that the fixed values should be merged with the existing values. The value overwrite indicates that the existing values are to be overwritten by the fixed values. The value preserve indicates that no changes should be made. The default value is merge.

A `fixedAttribute` element contains one or more `fixedValue` XML element, which specify the fixed LDAP values.

Validating Updated SCIM Schema

The PingData SCIM extension is bundled with an XML Schema document, `resources.xsd`, which describes the structure of a `scim-resources.xml` resource configuration file. After updating the resource configuration file, you should confirm that its contents are well-formed and valid using a tool such as `xmllint`.

For example, you could validate your updated file as follows:

```
$ xmllint --noout --schema resources.xsd scim-resources.xml
scim-resources.xml validates
```

Mapping SCIM Resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. The `entryUUID` attribute, whose read-only value is assigned by the Directory Proxy Server, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership. The resource configuration allows the SCIM resource ID to be mapped to the LDAP entry DN. However, the entry DN does not meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN. Likewise, you can use the Identity Access API to change the value of an entry's RDN attribute, thereby triggering a MODDN operation.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Directory Proxy Server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin may be used by the Directory Proxy Server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see "Working with the UID Unique Attribute Plug-in" in the PingDirectory Server Administration Guide.

Note: Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the `<resourceIDMapping>` Element".

Using Pre-defined Transformations

Transformations are required to change SCIM data types to LDAP syntax values. The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation`. Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation`. Transforms SCIM dateTime data type values to LDAP Generalized Time syntax values and vice-versa.
- `com.unboundid.scim.ldap.PostalAddressTransformation`. Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the $ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.
- `com.unboundid.scim.ldap.TelephoneNumberTransformation`. Transforms LDAP Telephone Number syntax (E.123) to RFC3966 format and vice-versa.

You can also write your own transformations using the SCIM API described in the following section.

Mapping LDAP Entries to SCIM Using the SCIM-LDAP API

In addition to the SCIM SDK, PingData provides a library called SCIM-LDAP, which provides facilities for writing custom transformations and more advanced mapping.

You can add the SCIM-LDAP library to your project using the following dependency:

```
<dependency>
    <groupId>com.unboundid.product.scim</groupId>
    <artifactId>scim-ldap</artifactId>
    <version>1.5.0</version>
</dependency>
```

Create your custom transformation by extending the `com.unboundid.scim.ldap.Transformation` class. Place your custom transformation class in a jar file in the server's `lib` directory.

Note: The Identity Access API automatically maps LDAP attribute syntaxes to the appropriate SCIM attribute types. For example, an LDAP DirectoryString is automatically mapped to a SCIM string.

SCIM Authentication

SCIM requests to the LDAP endpoints will support HTTP Basic Authentication and OAuth2 Authentication using a bearer token. There is existing support for this feature in the Directory Server and the Directory Proxy Server using the OAuthTokenHandler API (i.e., via a Server SDK extension, which requires some technical work to implement).

Note that our implementation only supports the HTTP Authorization header for this purpose; we do not support the form-encoded body parameter or URI query parameter mechanisms for specifying the credentials or bearer token.

SCIM Logging

The Directory Proxy Server already provides a detailed HTTP log publisher to capture the SCIM and HTTP request details. To be able to correlate this data to the internal LDAP operations that are invoked behind the scenes, the Access Log Publisher will use "`origin=scim`" in access log messages that are generated by the SCIM servlet.

For example, you will see a message for operations invoked by replication:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191
origin="replication" dn="uid=user.3,ou=people,dc=example,dc=com"
```

Likewise for SCIM messages, you will see a message like this:

```
[30/Oct/2012:18:45:10.490 -0500] MODFIY REQUEST conn=-3 op=190 msgID=191
origin="scim" dn="uid=user.3,ou=people,dc=example,dc=com"
```

SCIM Monitoring

There are two facilities that can be used to monitor the SCIM activity in the server.

- **HTTPConnectionHandlerStatisticsMonitorProvider** -- Provides statistics straight about total and average active connections, requests per connection, connection duration, processing time, invocation count, etc.
- **SCIMServletMonitorProvider** -- Provides high level statistics about request methods (POST, PUT, GET, etc.), content types (JSON, XML), and response codes, for example, "`user-patch-404:26`".

The LDAP object class endpoints are treated as their own resource types, so that for requests using the Identity Access API, there will be statistics, such as `person-get-200` and `inetorgperson-post-401`.

Configuring the Identity Access API

Once you have run the `<server-root>/config/scim-config-ds.dsconfig` script, the resources defined in the `scim-resources.xml` will be available as well as the Identity Access API. However, to allow SCIM access to the raw LDAP data, you must set a combination of configuration properties on the SCIM Servlet Extension using the `dsconfig` tool.

- **include-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will be exposed. The object class used here will be the one that clients need to use when referencing Identity Access API resources. This property allows the special value "*" to allow all object classes. If "*" is used, then the SCIM servlet uses the same case used in the Directory Proxy Server LDAP Schema.
- **exclude-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will not be exposed. When this property is specified, all object classes will be exposed except those in this list.
- **include-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will be exposed. If specified, only entries under these base DNs will be accessible. No parent-child relationships in the DNs are allowed here.
- **exclude-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will not be exposed. If specified, entries under these base DNs will not be accessible. No parent-child relationships in the DNs are allowed here.

Using a combination of these properties, SCIM endpoints will be available for all included object classes, just as if they were SCIM Resources defined in the `scim-resources.xml` file.

To Configure the Identity Access API

1. Ensure that you have run the `scim-config-ds.dsconfig` script to configure the SCIM interface. Be sure to enable the entryDN virtual attribute. See the Configure SCIM section for more information.
2. Set a combination of properties to allow the SCIM clients access to the raw LDAP data: `include-ldap-objectclass`, `exclude-ldap-objectclass`, `include-ldap-base-dn`, or `exclude-ldap-base-dn`.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM --set 'include-ldap-objectclass:*' \
  --set include-ldap-base-dn:ou=People,dc=example,dc=com
```

The SCIM clients now have access to the raw LDAP data via LDAP object class-based resources as well as core SCIM resources as defined in the `scim.resource.xml` file.

To Disable Core SCIM Resources

1. Open the `config/scim-resources.xml` file, and comment out or remove the `<resource>` elements that you would like to disable.
2. Disable and re-enable the HTTP Connection Handler, or restart the server to make the changes take effect. In general, changing the `scim-resources.xml` file requires a HTTP Connection Handler restart or server restart.

Note: When making other changes to the SCIM configuration by modifying the SCIM HTTP Servlet Extension using `dsconfig`, the changes take effect immediately without any restart required.

To Verify the Identity Access API Configuration

- Perform a curl request to verify the Identity Access API configuration.

```
$ curl -k -u "cn=directory manager:password" \
  -H "Accept: application/json" \
  "https://example.com/top/56c9fd6b-f870-35ef-9959-691c783b7318?
     attributes=entryDN,uid,givenName,sn,entryUUID"
     {"schemas":
["urn:scim:schemas:core:1.0","urn:unboundid:schemas:scim:ldap:1.0"],
      "id":"56c9fd6b-f870-35ef-9959-691c783b7318",
      "meta":{"lastModified":"2013-01-11T23:38:26.489Z",
      "location":"https://example.com:443/v1/top/56c9fd6b-
f870-35ef-9959-691c783b7318"},
      "urn:unboundid:schemas:scim:ldap:1.0":{"givenName":["Rufus"],"uid":
["user.1"],
      "sn":["Firefly"],"entryUUID":["56c9fd6b-f870-35ef-9959-691c783b7318"],
      "entrydn":"uid=user.1,ou=people,dc=example,dc=com"}}
```

Monitoring the SCIM Servlet Extension

The SCIM SDK provides a command-line tool, `scim-query-rate`, that measures the SCIM query performance for your extension. The SCIM extension also exposes monitoring information for each SCIM resource, such as the number of successful operations per request, the number of failed operations per request, the number of operations with XML or JSON to and from the client. Finally, the Directory Proxy Server automatically logs SCIM-initiated LDAP operations to the default File-based Access Logger. These operations will have an `origin='scim'` attribute to distinguish them from operations initiated by LDAP clients. You can also create custom logger or request criteria objects that can track incoming HTTP requests, which the SCIM extension rewrites as internal LDAP operations.

Testing SCIM Query Performance

You can use the `scim-query-rate` tool, provided in the SCIM SDK, to test query performance, by performing repeated resource queries against the SCIM server.

The `scim-query-rate` tool performs searches using a query filter or can request resources by ID. For example, you can test performance by using a filter to query randomly across a set of one million users with eight concurrent threads. The user resources returned to the client in this example is in XML format and includes the `userName` and `name` attributes.

```
scim-query-rate --hostname server.example.com --port 80 \
--authID admin --authPassword password --xml \
--filter 'userName eq "user.[1-1000000]"' --attribute userName \
--attribute name --numThreads 8
```

You can request resources by specifying a resource ID pattern using the --resourceID argument as follows:

```
scim-query-rate --hostname server.example.com --port 443 \
--authID admin --authPassword password --useSSL --trustAll\
--resourceName User \
--resourceID 'uid=user.[1-150000],ou=people,dc=example,dc=com'
```

The `scim-query-rate` tool reports the error "`java.net.SocketException: Too many open files`" if the open file limit is too low. You can increase the open file limit to increase the number of file descriptors.

Monitoring Resources Using the SCIM Extension

The monitor provider exposes the following information for each resource:

Number of successful operations per request type (such as GET, PUT, and POST).

Number of failed operations and their error codes per request type.

Number of operations with XML or JSON from client.

Number of operations that sent XML or JSON to client.

In addition to the information about the user-defined resources, monitoring information is also generated for the schema, service provider configuration, and monitor resources. The attributes of the monitor entry are formatted as follows:

```
{resource name}-resource-{request type}-{successful or error status code}
```

You can search for one of these monitor providers using an `ldapsearch` such as the following:

```
$ bin/ldapsearch --port 1389 bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN cn=monitor \
  --searchScope sub "(objectclass=scim-servlet-monitor-entry)"
```

For example, the following monitor output was produced by a test environment with three distinct SCIM servlet instances, Aleph, Beth, and Gimel. Note that the first instance has a custom resource type called host.

```
$ bin/ldapsearch --baseDN cn=monitor \
  '(objectClass=scim-servlet-monitor-entry)'
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Aleph)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Aleph)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
user-resource-delete-successful: 1
user-resource-put-content-xml: 27
user-resource-query-response-json: 3229836
user-resource-put-403: 5
user-resource-put-content-json: 2
user-resource-get-401: 1
user-resource-put-response-json: 23
user-resource-get-response-json: 5
user-resource-get-response-xml: 7
user-resource-put-400: 2
user-resource-query-401: 1141028
user-resource-post-content-json: 1
user-resource-put-successful: 22
user-resource-post-successful: 1
user-resource-delete-404: 1
user-resource-query-successful: 2088808
user-resource-get-successful: 10
user-resource-put-response-xml: 6
user-resource-get-404: 1
user-resource-delete-401: 1
user-resource-post-response-json: 1
```

```
host-resource-query-successful: 5773268
host-resource-query-response-json: 11576313
host-resource-query-400: 3
host-resource-query-response-xml: 5
host-resource-query-401: 5788152
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Beth)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Beth)
version: 1.2.0
build: 20120105174457Z
revision: 820
serviceproviderconfig-resource-get-successful: 3
serviceproviderconfig-resource-get-response-json: 2
serviceproviderconfig-resource-get-response-xml: 1
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
group-resource-query-successful: 245214
group-resource-query-response-json: 517841
group-resource-query-400: 13711
group-resource-query-401: 258916
user-resource-query-response-json: 107876
user-resource-query-400: 8288
user-resource-get-400: 33
user-resource-get-response-json: 1041
user-resource-get-successful: 2011
user-resource-query-successful: 45650
user-resource-get-response-xml: 1003
user-resource-query-401: 53938
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Gimel)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Gimel)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 1
schema-resource-query-401: 1
schema-resource-query-response-json: 2
user-resource-query-successful: 65
user-resource-get-successful: 4
user-resource-get-response-json: 6
user-resource-query-response-json: 132
user-resource-get-404: 2
user-resource-query-401: 67
```

About the HTTP Log Publishers

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.
- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.
- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as "'['dd/MM/yyyy:HH:mm:ss Z']'"
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information than the common log format in a format that is familiar to administrators who use the File-Based Access Log Publisher.

The HTTP Detailed Access Log Publisher generates log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-
38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-
d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the `dsconfig` command-line tool help.

Chapter 12

Managing Server SDK Extensions

Topics:

The PingDirectoryProxy Server provides support for any custom extensions that you create using the Server SDK. This chapter summarizes the various features and extensions that can be developed using the Server SDK.

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Proxy Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Available Types of Extensions

The Server SDK provides support for creating a number of different types of extensions for Ping Identity Server Products, including the PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. Some of those extensions include:

Cross-Product Extensions

Access Loggers
Alert Handlers
Error Loggers
Key Manager Providers
Monitor Providers
Trust Manager Providers
OAuth Token Handlers
Manage Extension Plugins

PingDirectory Server Extensions

Certificate Mappers
Change Subscription Handlers
Extended Operation Handlers
Identity Mappers
Password Generators
Password Storage Schemes
Password Validators
Plugins
Tasks
Virtual Attribute Providers

PingDirectoryProxy Server Extensions

LDAP Health Checks
Placement Algorithms
Proxy Transformations

PingDataSync Server Extensions

JDBC Sync Sources
JDBC Sync Destinations
LDAP Sync Source Plugins
LDAP Sync Destination Plugins
Sync SourcesSync Destinations

- Sync Pipe Plugins

For more information on the Server SDK, see the documentation available in the SDK build.

Chapter 13

Command-Line Tools

Topics:

The PingDirectoryProxy Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

This chapter presents the following topics:

Using the Help Option

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. You can view detailed argument options and examples by typing --help with the command.

```
bin/dsconfig --help
```

For those utilities that support additional subcommands (for example, `dsconfig`), you can get a list of the subcommands by typing --help-subcommands.

```
bin/dsconfig --help-subcommands
```

You can also get more detailed subcommand information by typing --help with the specific subcommand.

```
bin/dsconfig list-log-publishers --help
```

Note: For detailed information and examples of the command-line tools, see the *Ping Identity Directory Proxy Server Command-Line Tool Reference*.

Available Command-Line Utilities

The Directory Proxy Server provides the following command-line utilities, which can be run directly in interactive, non-interactive, or script mode.

Table 13: Command-Line Utilities

Command-Line Tools	Description
authrate	Perform repeated authentications against an LDAP directory server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more Directory Proxy Server backends. This utility also supports the use of a properties file to pass predefined command-line arguments. See Managing the tools.properties File for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a ZIP archive that can be sent to a technical support representative.
create-initial-proxy-config	Create an inital Directory Proxy Server configuration.
create-rc-script	Create an Run Control (RC) script that may be used to start, stop, and restart the Directory Proxy Server on UNIX-based systems.
create-recurring-task	Create a recurring task to run on the server. Tasks can be created for backups, LDIF exports, a statically defined task, or a third-party task.
create-recurring-task-chain	Create a chain of recurring tasks to run on the server.
dsconfig	View and edit the Directory Proxy Server configuration.
dsjavaproperties	Configure the JVM arguments used to run the Directory Proxy Server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and *JAVA_HOME* environment variable.

Command-Line Tools	Description
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Directory Server.
enter-lockdown-mode	Request that the Directory Proxy Server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege.
global-index-size	Estimates the size in memory of one or more global indexes from the actual number of keys, the configured number of keys and the average key size.
ldap-diff	Compare the contents of two LDAP directory server servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform LDAP compare operations in the Directory Proxy Server.
ldapdelete	Perform LDAP delete operations in the Directory Proxy Server.
ldapmodify	Perform LDAP modify, add, delete, and modify DN operations in the Directory Proxy Server.
ldappasswordmodify	Perform LDAP password modify operations in the Directory Proxy Server.
ldapsearch	Perform LDAP search operations in the Directory Proxy Server.
ldif-diff	Compare the contents of two LDIF files, the output being an LDIF file needed to bring the source file in sync with the target.
ldifmodify	Apply a set of modify, add, and delete operations against data in an LDIF file.
ldifsearch	Perform search operations against data in an LDIF file.
leave-lockdown-mode	Request that the Directory Proxy Server leave lockdown mode and resume normal operation.
list-backends	List the backends and base DNs configured in the Directory Proxy Server.
make-ldif	Generate LDIF data based on a definition in a template file.
manage-extension	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the PingDirectoryProxy Server. Extension bundles are installed from a zip archive or file system directory. The Directory Proxy Server will be restarted if running to activate the extension(s).
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Directory Proxy Server.
modrate	Perform repeated modifications against an LDAP directory server.
move-subtree	Move a subtree entries or a single entry from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
prepare-external-server	Prepare and a directory server for communication.
profile-viewer	View information in data files captured by the Directory Proxy Server profiler.
reload-index	Reload the contents of the global index.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Directory Proxy Server backend.
remove-defunct-server	Remove a server from this server's topology.
restore	Restore a backup of the Directory Proxy Server backend.

Command-Line Tools	Description
revert-update	Returns a server to the version before the last update was performed.
review-license	Review and/or indicate your acceptance of the product license.
scramble-ldif	Obscure the contents of a specified set of attributes in an LDIF file.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
search-rate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Directory Proxy Server process.
setup	Perform the initial setup for the Directory Proxy Server instance.
start-server	Start the Directory Proxy Server.
status	Display basic server information.
stop-server	Stop or restart the Directory Proxy Server.
subtree-accessibility	List or update the a set of subtree accessibility restrictions defined in the Directory Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
uninstall	Uninstall the Directory Proxy Server.
update	Update the Directory Proxy Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the `update` tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the `revert-update` tool for information on reverting an update.
validate-ldif	Validate the contents of an LDIF file against the server schema.

Managing the tools.properties File

The PingDirectoryProxy Server supports the use of a tools properties file that simplifies command-line invocations by reading in a set of arguments for each tool from a text file. Each property is in the form of name/value pairs that define predetermined values for a tool's arguments. Properties files are convenient when quickly testing the Directory Proxy Server in multiple environments.

The Directory Proxy Server supports two types of properties file: default properties files that can be applied to all command-line utilities or tool-specific properties file that can be specified using the --propertiesFilePath option. You can override all of the Directory Proxy Server's command-line utilities with a properties file using the `config/tools.properties` file.

Creating a Tools Properties File

You can create a properties file with a text editor by specifying each argument, or option, using standard Java properties file format (name=value). For example, you can create a simple properties file that define a set of LDAP connection parameters as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
baseDN=dc=example,dc=com
```

Next, you can specify the location of the file using the --propertiesFilePath /path/to/ File option with the command-line tool. For example, if you save the previous properties file as `bin/mytool.properties`, you can specify the path to the properties file with `ldapsearch` as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/mytools.properties "(objectclass=*)"
```

Properties files do not allow quotation marks of any kind around values. Any spaces or special characters should be escaped. For example,

```
bindDN=cn=QA\ Managers,ou=groups,dc=example,dc=com
```

The following is not allowed as it contains quotation marks:

```
bindDN=cn="QA Managers,ou=groups,dc=example,dc=com"
```

Tool-Specific Properties

The Directory Proxy Server also supports properties for specific tool options using the format: `tool.option=value`. Tool-specific options have precedence over general options. For example, the following properties file uses `ldapsearch.port=2389` for `ldapsearch` requests by the client. All other tools that use the properties file uses `port=1389`.

```
hostname=server1.example.com
port=1389
ldapsearch.port=2389
bindDN=cn=Directory\ Manager
```

Another example using the `dsconfig` configuration tool is as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
dsconfig.bindPasswordFile=/ds/config/password
```

👉 **Note:** The `.bindPasswordFile` property requires an absolute path. If you were to specify `~/ds/config/password`, where ~ refers to the home directory, the server does not expand the ~ value when read from the properties file.

Specifying Default Properties Files

The Directory Proxy Server provides a default properties files that apply to all command-line utilities used in client requests. A default properties file, `tools.properties`, is located in the `<server-root>/config` directory.

If you place a custom properties file that has a different filename as `tools.properties` in this default location, you need to specify the path using the --propertiesFilePath option. If you make changes to the `tools.properties` file, you do not need the --propertiesFilePath option. See the examples in the next section.

Evaluation Order Summary

The Directory Proxy Server uses the following evaluation ordering to determine options for a given command-line utility:

- All options used with a utility on the command line takes precedence over any options in any properties file.
- If the --propertiesFilePath option is used with no other options, the Directory Proxy Server takes its options from the specified properties file.

- If no options are used on the command line including the --propertiesFilePath option (and --noPropertiesFile), the Directory Proxy Server searches for the `tools.properties` file at `<server-root>`
- If no default properties file is found and a required option is missing, the tool generates an error.
- Tool-specific properties (for example, `ldapsearch.port=3389`) have precedence over general properties (for example, `port=1389`).

Evaluation Order Example

Given the following properties file that is saved as `<server-root>/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

The Directory Proxy Server locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (port and baseDN). The command uses the `bindDN` and `bindPassword` arguments specified in the properties file.

   ```
   $ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
     --propertiesFilePath bin/tools.properties "(objectclass=*)"
   ```
2. Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the Directory Proxy Server uses the specified properties file as follows:

   ```
   $ bin/ldapsearch --propertiesFilePath bin/tools.properties \
     "(objectclass=*)"
   ```
3. If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, the Directory Proxy Server attempts to locate any default `tools.properties` file in the following location:

   ```
   <server-root>/config/tools.properties
   ```

   Assume that you move your tools.properties file from `<server-root>/bin` to the `<server-root>/config` directory. You can then run your tools as follows:

   ```
   $ bin/ldapsearch "(objectclass=*)"
   ```

   The Directory Proxy Server can be configured so that it does not search for a properties file by using the --noPropertiesFile option. This options tells the Directory Proxy Server to use only those options specified on the command line. The --propertiesFilePath and --noPropertiesFile options are mutually exclusive and cannot be used together.
4. If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Running Task-based Utilities

The Directory Proxy Server has a Tasks subsystem that allows you to schedule basic operations, such as backup, restore, bin/start-server, bin/start-server and others. All task-based utilities require the --task option that explicitly indicates the utility is intended to run as a task rather than in offline mode. The following table shows the arguments that can be used for task-based operations:

Table 14: Task-based Utilities

Option	Description
--task	Indicates that the tool is invoked as a task. The --task argument is required. If a tool is invoked as a task without this --task argument, then a warning message will be displayed stating that it must be used. If the --task argument is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message will be displayed and the tool will exit with an error.
--start	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation starts when scheduled as a server task. A value of '0' causes the task to be scheduled for immediate execution. When this option is used, the operation is scheduled to start at the specified time, after which this utility will exit immediately.
--dependency	Specifies the ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution. This option can be used multiple times in a single command.
--failedDependencyAction	Specifies the action this task will take should one of its dependent tasks fail. The value must be one of the following: `PROCESS`, `CANCEL`, `DISABLE`. If not specified, the default value is `CANCEL`. This option can be used multiple times in a single command.
--completionNotify	Specifies the email address of a recipient to be notified when the task completes. This option can be used multiple times in a single command.
--errorNotify	Specifies the email address of a recipient to be notified if an error occurs when this task executes. This option can be used multiple times in a single command.

PingDirectory™

Release 7.0

Consent Solution Guide

Notice

PingDirectory™ Product Documentation

Trademarks

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Contents

Chapter 1

Introduction to the Consent Service and Consent API

Topics:

Companies gain loyalty and trust when they offer transparency and control to their users and customers regarding the personal data that is collected, processed, or shared. In Europe, the General Data Protection Regulation (GDPR) was designed specifically for allowing companies to collect and use valuable data about its users, while protecting the rights of citizens to control what is collected and used. To support the collection and end-user control of personal data, PingDirectory Server includes schema and REST APIs that provide the ability to collect fine-grained data authorizations (consents), from users and customers.

Consent Service overview

The Consent Service is an HTTP-based REST API hosted by the PingDirectory Server or PingDirectoryProxy Server. The service enables the collection of consent from application users, the enforcement of consent, a user's management of his or her consent, and auditing of consent actions. Enterprises can integrate these features into their applications to give users transparency and control of their data privacy.

For the purpose of this document, the following terms are used:

Table 1: Consent Terms

Term	Meaning
Consent definition	The terms of the fine-grained contract, which describes the data that can be processed or shared, and a purpose for processing or sharing the data. The consent definition is stored in the server configuration.
Consent localization	A child object of a definition that contains versioned, localized text for the consent definition, to be used when prompting an individual. This is stored in the server configuration.
Consent record	A record of a consent interaction with a user. Consent records are stored in the directory tree.
Subject	The individual whose data can be collected, processed, or shared.
Actor	The individual who granted/denied/revoked consent. This is usually the same as the subject.
Audience	The entity, application, or service that is granted or denied access to a subject's data for a specific purpose.

Consent API overview

The PingDirectory Server and PingDirectoryProxy Server provide a REST API for managing individuals' consent to handle their data. This can be used as a component of a larger solution, such as a GDPR compliance system or the PingDataGovernance Server Open Banking Account Requests API.

The PingDirectory Server Consent API enables authorization services:

- to capture user consent for sharing or processing data
- to confirm that consent to share or process data has been granted
- for individuals to manage the consent that they have granted.

Detailed API documentation can be found on the Ping Identity website.

How consents are collected

User consent is collected by creating a consent record through the Consent API. In most cases, the Consent API client uses consent localization data to construct an approval prompt to display to the user. This prompt should include text describing what data is collected and for what purpose, allowing the user to make an informed decision about the value of sharing his or her data.

For example, a web application needing to collect consent for a user's browsing behavior would use the Consent API to look up the localizations for the `browsing-behavior` consent definition. It would select the localization appropriate for the user and use data from the localization resource to construct a consent prompt for display to the user. After the user is prompted and makes a decision, the client could store the decision by creating a new consent record through the Consent API.

How consents are enforced

The Consent Service can be used as a data source for making access control decisions. If a particular data usage scenario requires consent, then the application or service needing to access or process that data must not be able to use the data unless the user has provided consent. The entity that performs this consent check may be the application itself or some other service.

To perform a consent check, the Consent API client must be able to correlate a data access request type with a consent definition. For example, if a web application needs to collect a user's browsing behavior, this data collection scenario might be represented by a consent definition called `browsing-behavior`. The application would check for an existing consent grant by searching the Consent API for a consent record that matches the user and the `browsing-behavior` consent definition. If a match is found, then the application can proceed. If a match is not found, the application must collect consent from the user.

How applications use the Consent API

The following example illustrates both consent capture and consent enforcement. This example follows a user's journey on a website during which the company must gather consent to track the user's browsing behavior:

1. A user launches the company's application and authenticates. The application wants to record the page visit, but first it must check if the user has granted consent to do so.
2. The application makes a call to the Consent API to determine if the `browsing-behavior` consent record exists for this user, and whether consent been granted.
3. The API returns a result indicating that no consent record exists. The application must prompt the user for his or her consent. The application calls the Consent API to retrieve the localization for the `browsing-behavior` consent, which includes the language that the application uses to produce a prompt for the user.
4. After the user makes a decision, the application stores the user's decision by creating a new consent record. This is through a call to the Consent API.
5. Later, the user visits another page in the company's site. The application wants to record the page visit, and again checks whether the user has granted consent to do so.
6. The application makes a call to the Consent API to get the `browsing-behavior` consent record for this user.
7. If the user's consent record agrees to have the company track his or her browsing behavior, the application can then make the appropriate calls to track browsing behavior. This is consent enforcement.

Chapter

2

Consent Service configuration

Topics:

This section provides details for installing and configuring the components on which the Consent Service relies. Refer to the PingDirectory Server Administration Guide for detailed configuration information.

Configuration overview

The Consent Service is not enabled by default. The setup and configuration process varies depending on the following factors:

- Whether client applications will allow an individual to self-manage consents.
- Whether some or all client applications will be privileged, with the ability to manage all consents.
- The HTTP authentication method used by client applications.
- Whether consent records exist in the same directory as user entries.

Example configuration scenarios

The following client application scenarios are available for determining how the Consent Service should be configured to meet your business needs.

Directly managed consents

In this scenario, one or more client applications provide provide an interface for individuals to directly manage their own consent records. These applications can only manage consents for the currently authenticated user. In addition, there is also a client application for consent administrators. An OAuth 2 authorization server grants access tokens that the applications uses to access the Consent API.

Configuration for this scenario includes:

1. Configure an OAuth 2 authorization server to issue a `urn:pingdirectory:consent` scope to individuals and a `urn:pingdirectory:consent_admin` scope to consent administrators.
2. Create an identity mapper to map subject identifiers used by the authorization server to LDAP DNs used by the PingDirectory Server.
3. Configre an access token validator to validate tokens issued by the OAuth 2 authorization server.
4. Configure the Consent HTTP Servlet Extension to disable HTTP basic authentication and restart the HTTPS Connection Handler.
5. Configure the Consent Service to use the OAuth scopes, token validator, and identity mapper.

Indirectly managed consents (basic authentication)

In this scenario, an application uses a privileged service account to manage its users' consents. The application's privileged account can access any consent record, which gives the application the ability to perform operations that an individual user cannot. The following include steps the setup needed for the PingDataGovernance Server's Open Banking Account Requests service to use the Consent Service as its backend.

Configuration for this scenario includes:

1. Create a service account for the application.
2. Configure the Consent HTTP Servlet Extension to enable HTTP basic authentication and restart the HTTPS Connection Handler.
3. Create an identity mapper to map consent record subject and actor attribute values to LDAP DNs. This is optional.
4. Configure the Consent Service to use the application's service account, and optionally the identity mapper.

Set up with the configuration scripts

PingDirectory Server includes two configuration scripts that can serve as the starting point for setting up the Consent Service. Both scripts must be carefully reviewed and updated to support your client application scenarios and business needs.

- `consent-service-base-entries.ldif` - This LDIF script can be imported to create the base DN where consent records will be stored.
- `consent-service-cfg.dsconfig` - This script can be imported to configure and enable the Consent Service.

Both are located in the `/resource/consent/` directory of the PingDirectory Server server root.

Basic configuration with the `consent-service-base-entries.ldif` file includes:

1. Edit the LDIF script and change the location of where consent records will be stored.
2. Import the LDIF script using the `ldapmodify`command, such as:

```
$ bin/ldapmodify --defaultAdd \
  --filename consent-service-base-entries.ldif
```

Basic configuration with the `consent-service-cfg.dsconfig` file includes:

1. Search for `CHANGE-ME` and replace values.
2. Review configuration commands and make additional changes to match existing Ping environment parameters, application scenarios, and business needs.
3. Impost the script with the `dsconfig` command, such as:

```
$ bin/dsconfig --no-prompt \
  --batch-file consent-service-cfg.dsconfig
```

Setup in a replicated PingDirectory Server environment

Running the Consent Service setup script requires special consideration in an environment that includes replicated PingDirectory Servers. If possible, setup the Consent Service after replication is enabled for the PingDirectory Servers. See the *PingDirectory Server Administration Guide* for details about server replication.

Set up Consent Service after replication is enabled

Complete the following steps if replication is already enabled for PingDirectory Servers.

1. If needed, configure the PingDirectory Servers to use a configuration group called "all-servers." This will ensure that configuration changes are applied to all servers in a topology.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:all-servers
```

2. Run the Consent Service setup script.

```
$ bin/dsconfig --no-prompt \
  --batch-file resource/consent/consent-service-cfg.dsconfig
```

Set up Consent Service before replication is enabled

If you have already set up the Consent Service on a standalone PingDirectory Server, perform the following the steps before enabling replication. In this example, "DS1" is the original PingDirectory Server, and "DS2" is the second server that will be added as a replica.

1. Run the `config-diff` command without arguments on DS1 to produce a batch file that contains configuration changes that will be applied to DS2.

```
$ bin/config-diff > config-changes.dsconfig
```

2. Apply the `config-changes.dsconfig` file to DS2.

   ```
   $ bin/dsconfig --no-prompt \
     --batch-file config-changes.dsconfig \
     --applyChangeTo single-server
   ```

3. Restart DS2.
4. Enable replication between the two servers.

Configuration reference

There are many configuration options for the Consent Sevice and application integration. The configuration scripts included with the PingDirectory Server provide a starting point. Additional detailed information about the Consent Service properties and configuration is provided as reference.

General Consent Service configuration

The Consent Service configuration is used to control authorization behavior and determines where consent records are stored in the PingDirectory Server. The service properties are configured with the `dsconfig set-consent-service-prop` command. The consent service configuration script configures the consent service properties as follows:

```
$ bin/dsconfig set-consent-service-prop \
  --set enabled:true \
  --set base-dn:ou=consents,dc=example,dc=com \
  --set "bind-dn:cn=consent service account" \
  --set unprivileged-consent-scope:urn:pingdirectory:consent \
  --set privileged-consent-scope:urn:pingdirectory:consent_admin \
  --set "consent-record-identity-mapper:User ID Identity Mapper"
```

The following are Consent Service properties.

Table 2: Consent Service properties

Property	Description	Required to enable service
`enabled`	If set to true, enables the Consent Service for handling client requests.	Yes
`base-dn`	Specifies a container DN for consent record entries.	Yes
`bind-dn`	Specifies an internal service account used by the Consent Service to perform LDAP operations.	Yes
`service-account-dn`	Specifies one or more DNs of requesters that will be considered privileged when using basic authentication. If not defined, a requester will only be considered privileged if it is mapped to a DN with the `bypass-acl` privilege. Optional.	No
`unprivileged-consent-scope`	Specifies the name of the scope required for bearer tokens representing unprivileged requesters.	Yes
`privileged-consent-scope`	Specifies the name of the scope required for bearer tokens representing privileged requesters.	Yes
`consent-record-identity-mapper`	Specifies one or more identity mappers used to map consent record `subject` and `actor` values to DNs. If not defined,	No

Property	Description	Required to enable service
	the Consent Service will not handle requests from unprivileged clients. Optional.	
`audience`	Specifies an `audience` claim value that the Consent Service will require to be present in bearer tokens that it accepts. Optional.	No

For the Consent Service to report itself as unavailable to clients, the following must be true:

- The Consent Service must be enabled.
- The Consent Service base DN must be configured and must exist.
- The internal service account must be configured and must exist.
- The internal service account must have the right to read, add, modify, and delete entries under the Consent Service base DN.

Create a container entry for consent records

Each consent record is a distinct entry in the PingDirectory Server, and the Consent Service requires that these entries be stored under a common base DN, defined by the `base-dn` property of the Consent Service configuration. The Consent Service LDIF file sets the base DN. Use these steps to choose a different location to store consent records.

1. To create the Consent Service base DN, open a text editor and save the following to the file `consent-service-base-dn.ldif`.

```
dn: ou=consents,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: consents
```

2. Use `ldapmodify` to add the entry.

```
$ bin/ldapmodify --defaultAdd --filename consent-service-base-dn.ldif
```

Create an internal service account

The Consent Service uses an internal LDAP connection to operate against consent records that are stored as LDAP entries. It authenticates this LDAP connection using a service account, which must be created and dedicated solely to the Consent Service.

The Consent Service configuration script configures the internal service account using a topology admin user. If needed, this can be changed to a root DN user or a user DN whose entry is in the user backend. In all cases, the service account should exist in every LDAP server in the topology.

This service account must have full read and write access to the Consent Service base DN, the ability to read users' `isMemberOf` attribute, and the right to use the following LDAP controls:

- IntermediateClientRequestControl (1.3.6.1.4.1.30221.2.5.2)
- NameWithEntryUUIDRequestControl (1.3.6.1.4.1.30221.2.5.44)
- RejectUnindexedSearchRequestControl (1.3.6.1.4.1.30221.2.5.54)
- PermissiveModifyRequestControl (1.2.840.113556.1.4.1413)
- PostReadRequestControl (1.3.6.1.1.13.2)

For more information about configuring access, see the *"Managing Access Contro"l* chapter of the PingDirectory Server Administration Guide.

- To ensure the correct access, create a user with the `bypass-acl` privilege. The following `dsconfig` command creates a topology admin user with the `bypass-acl` privilege. After this is created, set this user as the `bind-dn` for the Consent Service.

```
$ dsconfig create-topology-admin-user \
  --user-name "Consent Service Account" \
  --set "description:Consent API service account" \
  --set "alternate-bind-dn:cn=consent service account" \
  --set first-name:Consent \
  --set inherit-default-root-privileges:false \
  --set last-name:Service \
  --set password:CHANGE-ME \
  --set privilege:bypass-acl
```

- Because the `bypass-acl` privilege grants a broad level of access, you may not want to grant this privilege to the Consent Service account. If desired, add the following ACI to enable a targetted set of functionality for the Consent Service. The following example grants this access to the DN `cn=consent service account` using global ACIs:

```
# Grant access to the consent record base DN ou=consents,dc=example,dc=com
dsconfig set-access-control-handler-prop --add 'global-aci:(target="ldap:///
ou=consents,dc=example,dc=com")(targetattr="*||+")(version 3.0; acl "Consent
 Service account access to consent record data"; allow(all) userdn="ldap:///
cn=consent service account";)'

# Grant access to the LDAP request controls used by the Consent Service.
dsconfig set-access-control-handler-prop --add 'global-aci:
(targetcontrol="1.3.6.1.4.1.30221.2.5.2||1.3.6.1.4.1.30221.2.5.44||
1.3.6.1.4.1.30221.2.5.54||1.2.840.113556.1.4.1413||1.3.6.1.1.13.2")(version
 3.0; acl "Consent Service account access to selected controls"; allow
 (read) userdn="ldap:///cn=consent service account";)'

# Grant access to use the Effective Rights Control.
dsconfig set-access-control-handler-prop --add 'global-aci:
(targetcontrol="1.3.6.1.4.1.42.2.27.9.5.2")(version 3.0; acl "Consent
 Service account access to Get Effective Rights control"; allow (read)
 userdn="ldap:///cn=consent service account";)'
```

Configure an identity mapper

The Consent Service uses identity mappers to map requester identities, subject values, and actor values to DNs. An identity mapper takes a user identifier string and correlates the identifier with the DN of a user entry. The PingDirectory Server provides four different types of identity mappers.

Table 3: Identity mappers

Identity mapper type	Description
Exact match identity mapper	Maps a user identifier to a DN by searching for an entry with an attribute that exactly matches the identifier.
Regular expression identity mapper	Similar to an exact match identity mapper, but allows a regular expression to be specified for more flexible matching.
Third-party identity mapper	A custom Java identity mapper implementation written using the Server SDK.
Groovy scripted identity mapper	A custom Groovy identity mapper implementation written using the Server SDK.

The Consent Service can be configured to use identity mappers for each of the following scenarios:

- Requesters authenticating using basic authentication - use the Consent HTTP Servlet Extension `identity-mapper` property to configure an identity mapper that takes the HTTP Basic authorization username string to find the corresponding user's identity in the PingDirectory Server.

- Requesters authenticating using bearer token authentication - use the Access Token Validator `identity-mapper` property to configure an identity mapper that takes the subject (or other claim value from the OAuth token) to find the corresponding user's identity in the PingDirectory Server.
- Consent record actor and subject values - use the Consent Service `consent-record-identity-mapper` property to configure an identity mapper that takes these consent record attribute values and uses them to find the corresponding users' identities in the PingDirectory Server.

The Consent Service configuration script configures a single identity mapper to be used for all three scenarios. The provided identity mapper searches by `uid`, `cn`, or `entryUUID` attributes under the base DNs `cn=config and ou=people,dc=example,dc=com`.

The following configuration provides an example of an identity mapper that will match a user identifier to an LDAP entry with the same value in its `uid` attribute:

```
$ bin/dsconfig create-identity-mapper --mapper-name "User ID Exact Match" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:uid
```

The following configuration shows another typical example, that of an identity mapper that will match a user identifier to an LDAP entry with the same value in its `entryUUID` attribute:

```
$ bin/dsconfig create-identity-mapper --mapper-name "EntryUUID Exact Match"
 \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:entryUUID
```

The last example creates an identity mapper that will match a user identifier to an LDAP entry with the same value in either its `uid`, `cn`, or `entryUUID` attribute. This identity mapper will also constrain its search to the `ou=people,dc=example,dc=com` and `cn=config` base DNs. (The `cn=config` base DN is not searched by default, and must be explicitly listed to be searched.)

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "User ID Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:uid \
  --set match-attribute:cn \
  --set match-attribute:entryUUID \
  --set match-base-dn:cn=config \
  --set match-base-dn:ou=people,dc=example,dc=com
```

Authentication methods

The Consent Service supports two HTTP authentication methods, which are both enabled by default:

- Basic authentication
- Bearer token authentication

The Consent servlet looks at the request's Authorization header to determine which authentication type is being used by the client.

With basic authentication, the client provides an encoded username/password pair in the HTTP Authorization request header. When the Consent Service receives a request using basic authentication, it maps the username credential to a DN using an identity mapper. This DN is designated the `auth DN` and is used to make subsequent authorization decisions. The Consent Service then performs an LDAP bind using the DN and password to determine if the request can be processed.

With bearer token authentication, the client provides an access token in the HTTP Authorization request header. The access token is always obtained by the client from an external OAuth 2 authorization server and encapsulates information ("claims") about a user identity, the client identity, and the requests that the client is authorized to make.

The PingDirectory Server must be configured to accept access tokens using one or both available access token validators:

- **PingFederate access token validator**. Supports access tokens issued by a PingFederate authorization server. This validator verifies an access token and discovers its claims by making a request to the PingFederate server's token introspection endpoint.
- **JWT access token validator**. Supports signed or encrypted JWT access tokens issued by an arbitrary authorization server. This validator checks an access token by cryptographically verifying the token's signature using a trusted public certificate. The token's claims are encoded in the token itself, so discovering the token's claims does not require an outgoing token introspection request.

The token validator uses its identity mapper to map the subject claim to a DN. This DN is designated the `auth DN` and is used along with the token's claims to make subsequent authorization decisions.

If the PingDirectory Server is configured with at least one access token validator, it will be used by the Consent Service. If the PingDirectory Server is configured with more than one access token validator, the validators are consulted in order until one is able to successfully authenticate the request.

If the PingDirectory Server is configured with multiple access token validators, but only one should be used by the Consent Service, the access token validator can be configured by setting the `access-token-validator` property of the Consent HTTP Servlet Extension.

👉 **Note:** Configuring an access token validator for the Consent Service requires information from the authorization server configuration:

- The values that the authorization server sets for `subject` claims must be mappable to a DN in the PingDirectory Server.
- The authorization server must be configured to authorize clients and grant scopes appropriately for privileged or unprivileged Consent API access.
- The authorization server must be configured to issue tokens with scopes corresponding to the Consent Service's `unprivileged-scope-name` and `privileged-scope-name` configuration.

Refer to the authorization server's documentation for guidance.

Configure basic authentication

Basic authentication is enabled by default, and the settings are configured in the Consent HTTP Servlet Extension configuration.

- Use the following command to disable basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set basic-auth-enabled:false
```

- Use the following command to enable basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set basic-auth-enabled:true
```

- Use the following command to configure an identity mapper for basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "identity-mapper:User ID Exact Match"
```

- All of these configuration changes require the Consent servlet to be reloaded before they can take effect. Use the following commands to restart the connection handler that hosts the Consent servlet.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:false
```

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true
```

Configure bearer token authentication

- The following is an example access token validator configured on the PingDirectory Server for a PingFederate server:

```
$ bin/dsconfig create-external-server \
  --server-name PingFederate \
  --type http \
  --set base-url:https://my-ping-federate-server:1443/
```

```
$ bin/dsconfig create-access-token-validator \
  --validator-name "PingFederate Token Validator" \
  --type ping-federate \
  --set enabled:true \
  --set "identity-mapper:User ID Exact Match" \
  --set authorization-server:PingFederate \
  --set client-id:id \
  --set client-secret:secret
```

- If more than one access token validator is configured on the PingDirectory Server, the Consent Service can be configured to use a single validator with the following command:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "access-token-validator:PingFederate Token Validator"
```

Configure Consent Service scopes

The Consent Service checks access tokens for a `subject` claim and uses an identity mapper to map the value to a DN, called the request DN or auth DN. If no request DN can be mapped, the request is rejected. In addition, the Consent Service will only accept an access token with a scope that it is configured to recognize.

- An unprivileged consent scope designates the requester as unprivileged. The scope's name is configured with the Consent Service's `unprivileged-consent-scope` property.
- A privileged consent scope designates the requester as privileged. This is configured using the Consent Service's `privileged-consent-scope` property.

The authorization server must also be configured to issue tokens with these scopes.

- The following example configures these scopes for the Consent Service.

```
$ bin/dsconfig set-consent-service-prop \
  --set privileged-users-group-dn:dn_name \
  --set unprivileged-consent-scope:consent \
  --set privileged-consent-scope:consent_admin
```

Authorization

The Consent Service's distinction between privileged and unprivileged requesters determines the type of operations that can be performed by requesters. During the authorization phase, the Consent servlet performs checks on both the bearer token claims (if present) and the `auth DN` to determine if the requester is privileged or unprivileged. These are summarized in the following table.

Table 4: Available operations per requester type

Requester type	Description	Access determined by	Can create consent records	Can update consent records	Can delete consent records
Unprivileged	Requesters with no authority to operate on consent records other than their own. Unprivileged requesters can only be used if the Consent Service is configured to map subject and actor DNs using the `consent-record-identity-mapper` property.	A requester is considered unprivileged if it does not meet any of the criteria for a privileged requester. If using bearer token authentication, the access token must include a scope named by the `unprivileged-consent-scope` property of the Consent Service configuration. Also, an unprivileged requester can only perform actions on consent records where the subject DN and actor DN match the requester DN.	Yes, if the requester DN matches the subject DN and actor DN.	Yes, if the requester DN matches the subject DN and actor DN.	No.
Privileged	A requester with the authority to perform any operation on any consent record.	When using basic authentication, a requester is considered privileged if the requester DN either has the `bypass-acl` privilege or is listed in the `service-account-dn` property of the Consent Service configuration. If using bearer token authentication, the access token must include a scope named by the `privileged-consent-scope` property of the Consent Service configuration.	Yes.	Yes.	Yes.

Bearer token check

If a bearer token was used, the following checks are performed:

- If the Consent Service's `audience` property is configured, the bearer token's audience claim must match the configured value.
- If the bearer token contains a scope matching the Consent Service's `privileged-scope-name` property, then the requester is considered privileged.
- If not, the bearer token must have a scope matching the Consent Service's `unprivileged-scope-name` property, and the requester is considered unprivileged.

Basic authentication check

If basic authentication is used, the following checks are performed:

- If the `auth DN` has the LDAP privilege `bypass-acl`, the requester is privileged.
- If the `auth DN` is listed in the Consent Service's `service-account-dn` property, the requester is privileged.
- If not, the requester is considered unprivileged.

Chapter 3

Manage consents

Topics:

This section describes the tasks required to support the collection and end-user control of personal data, and manage users' consents.

Overview of consent management

The full lifecycle of consent management goes beyond collecting the user's consent. First, the terms of each consent contract must be centrally managed. After collecting consent, the user will want to review previously granted consents and potentially revoke some. Finally, companies will need to be able to trace the history of updates to any consent in order to resolve a dispute or respond to audit.

Consent definitions and localizations

Companies will want to centrally manage the language used when prompting a user to give consent. This is key to ensuring a consistent user experience across multiple applications, such as mobile and web. The Consent Service requires one or more consent definitions to be defined in the PingDirectory Server configuration. Each consent definition represents the combination of:

- The data to be collected or shared.
- The purpose for collecting or sharing this data.

For example, a consent definition could represent user email addresses, used to deliver a third party's email newsletter. A consent definition could also represent access to a user's network-connected IoT device, which would be used for a home automation task controlled by a third party.

Each consent definition must have one or more localization. A localization is a versioned object consisting of the data that a Consent API client needs to prompt a user for consent. When a consent record is accepted or denied by a Consent Service client, it must include a reference to a consent definition, locale, and version.

Create consent definition and localization

- The following creates a consent definition and a localization for it.

  ```
  $ bin/dsconfig create-consent-definition \
    --definition-name email_newsletter \
    --set "display-name:Email newsletter"
  ```

  ```
  $ bin/dsconfig create-consent-definition-localization \
    --definition-name email_newsletter \
    --localization-name en-US \
    --set version:1.0 \
    --set "data-text:Your email address" \
    --set "purpose-text:To receive newsletter updates"
  ```
- The following example updates a localization and its version.

  ```
  $ bin/dsconfig set-consent-definition-localization-prop \
    --definition-name email_newsletter \
    --localization-name en-US \
    --set version:1.1 \
    --set "data-text:Your preferred email address"
  ```

Perform an audit on consents

Changes to Consent Service resources are tracked by one of two types of audit logs. For examples of configuring either type of log, see the `<server-root>/resource/consent-service-cfg.dsconfig` script bundled with the server or *Logging*. This example uses the Consent Trace Logger. It represents Consent Service change events using the same field names used by the Consent API.

Table 5: Log Publishers

Log publisher	Log publisher type	Description
Consent Trace Logger	file-based-trace	Records Consent Service events at the Consent API level. Change events are recorded using messages of type `audit`.
Consent LDAP Audit Logger	file-based-audit	Records data changes at the LDAP level. In combination with a Request Criteria configuration object, an LDAP audit logger can be configured to record changes to Consent Service resources only.

Trace logger keys for auditing

Trace logger audit messages consist of a timestamp, the message type (`CONSENT AUDIT`), and a set of key/value pairs. A subset of important keys are described in the following table.

Note: The keys used in trace log audit messages vary depending on the type of resource.

Table 6: Log Publishers

Trace logger key	Description
`requestID`	A server-specific HTTP request ID. This value can be correlated with messages produced by other loggers.
`resourceType`	The type of Consent Service resource that was changed. Possible values are `definition`, `localization`, or `consent`.
`changeType`	The type of change recorded by this message. Possible values are `create`, `update`, or `delete`.
`attrsAdded`	A comma-delimited list of the attributes that were added to the resource.
`attrsUpdated`	A comma-delimited list of the attributes that were modified on the resource.
`attrsDeleted`	A comma-delimited list of the attributes that were removed from the resource.
`requestDN`	The DN of the requester, which is available only when the resource type is `consent`.
`definitionID`	The consent definition ID. If the resource type is `definition`, this identifies the definition that was changed. If the resource type is `localization`, this identifies the parent definition. If the resource type is `consent`, this identifies the consent record's related definition.
`locale`	The locale. If the resource type is `localization`, this identifies the localization (in combination with the definition ID). If the resource type is `consent`, this identifies the related localization (combined with the definition ID).
`consentID`	The consent record ID, available only when the resource type is `consent`.
`subject`	The subject value, available only when the resource type is `consent`.
`subjectDN`	The subject's mapped LDAP DN, available only when the resource type is `consent`.
`actor`	The actor value, available only when the resource type is `consent`.

Trace logger key	Description
`actorDN`	The actor's mapped LDAP DN, available only when the resource type is `consent`.
`audience`	The audience value, available only when the resource type is `consent`.
`status`	The consent status. Possible values are `pending`, `accepted`, `denied`, `revoked`, and `restricted`. Only available when the resource type is `consent`.
`previousStatus`	The previous consent status, if applicable. Only available when the resource type is `consent`.
`msg`	A multiline value that includes the complete body of the changed resource. If the action is an update or a delete, the resource's body before the change will be included.

Perform an audit

Consent resource changes for particular entities (such as a specific user, or a specific consent definition) can be audited by searching the trace log using a combination of one of the message keys and the desired value. For example, if an individual's LDAP DN is known, then the `subjectDN` key can be used to construct a text search for any audit log messages containing that DN. Any matching log messages would constitute a history of that individual's consent activity.

Example new consent record

The following is a sample record. this audit log message provides important values in a parseable key/value format, but also includes the entirety of the new consent record.

```
[22/May/2018:18:02:42.584 -0500] CONSENT AUDIT requestID=57
 requestDN="uid=user.0,ou=people,
  dc=example,dc=com" consentID="6cff325b-e092-4094-b7f9-5a30864b0d24"
 subject="user.0" subjectDN="uid=user.0,
  ou=People,dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com" audience="client1"
  definitionID="cats" locale="en-US" status="accepted"
 attrsAdded="actor,audience,createdDate,dataText,subject,
  purposeText,definition,id,updatedDate,actorDN,status,subjectDN"
 changeType="create" resourceType="consent" msg="
New Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'accepted','subject':'user.0','subjectDN':'uid=user.0,

 ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=People,dc=exampl
  'client1','definition':{'id':'cats','version':'1.0','locale':'en-
US'},'dataText':'Collect data about your
  cats','purposeText':'To recommend cat food flavors that will satisfy and
 delight your feline companion',

 'createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:02:42.553Z'}"
```

Example updated consent record

This example shows the complete consent record before and after it was updated. With the `attrsUpdated`, `status`, and `previousStatus` keys, one can determine that the `status` changed from accepted to revoked.

```
[22/May/2018:18:05:08.660 -0500] CONSENT AUDIT requestID=59
 requestDN="uid=user.0,ou=people,
   dc=example,dc=com" consentID="6cff325b-e092-4094-b7f9-5a30864b0d24"
 subject="user.0" subjectDN="uid=user.0,
   ou=People,dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com"
   audience="client1" definitionID="cats" locale="en-US" status="revoked"
 previousStatus="accepted"
   attrsUpdated="status" changeType="update" resourceType="consent" msg="
Previous Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'accepted','subject':'user.0','subjectDN':'uid=user.0,

 ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=People,dc=exampl
  'audience':'client1','definition':
{'id':'cats','version':'1.0','locale':'en-US'},'dataText':'Collect
  data about your cats','purposeText':'To recommend cat food flavors that
 will satisfy and delight your
  feline
 companion','createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:02:42.
Updated Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'revoked','subject':'user.0','subjectDN':

 'uid=user.0,ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=Peop
  dc=com','audience':'client1','definition':
{'id':'cats','version':'1.0','locale':'en-US'},'dataText':
  'Collect data about your cats','purposeText':'To recommend cat food
 flavors that will satisfy and
  delight your feline
 companion','createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:05:08.
```

Example deleted consent record

This example shows that a consent record has been deleted, and the complete representation of the consent record prior to its deletion is provided.

```
[22/May/2018:18:06:35.071 -0500] CONSENT AUDIT requestID=61
 requestDN="cn=directory manager"
   consentID="6cff325b-e092-4094-b7f9-5a30864b0d24" subject="user.0"
 subjectDN="uid=user.0,ou=People,
   dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com" audience="client1"
   definitionID="cats" locale="en-US" status="revoked"
 previousStatus="revoked" attrsDeleted="actor,audience,

 createdDate,dataText,subject,purposeText,definition,id,updatedDate,actorDN,status,subje
 changeType="delete"
   resourceType="consent" msg="
Deleted Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'revoked','subject':'user.0','subjectDN':

 'uid=user.0,ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=Peop
   dc=example,dc=com','audience':'client1','definition':
{'id':'cats','version':'1.0','currentVersion':
   '1.0','locale':'en-US'},'dataText':'Collect data about your
 cats','purposeText':'To recommend cat food
   flavors that will satisfy and delight your feline
 companion','createdDate':'2018-05-22T23:02:42.553Z',
   'updatedDate':'2018-05-22T23:05:08.655Z'}"
```

Logging

The PingDirectory Server trace log publisher is used for logging events generated by HTTP service operations. The trace logger can be used to observe, debug, and audit consent requests.

Note: To create a log of consent audit events only, remove all message types except for `consent-message-type:audit`.

- The following example of creates a trace logger for all consent events, plus summaries of HTTP requests and responses.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "Consent Trace Logger" \
  --type file-based-trace \
  --set "description:Records Consent API operations" \
  --set enabled:true \
  --set consent-message-type:audit \
  --set consent-message-type:consent-created \
  --set consent-message-type:consent-deleted \
  --set consent-message-type:consent-retrieved \
  --set consent-message-type:consent-search \
  --set consent-message-type:consent-updated \
  --set consent-message-type:definition-created \
  --set consent-message-type:definition-deleted \
  --set consent-message-type:definition-retrieved \
  --set consent-message-type:definition-search \
  --set consent-message-type:definition-updated \
  --set consent-message-type:error \
  --set consent-message-type:localization-created \
  --set consent-message-type:localization-deleted \
  --set consent-message-type:localization-retrieved \
  --set consent-message-type:localization-search \
  --set consent-message-type:localization-updated \
  --set http-message-type:request \
  --set http-message-type:response \
  --set 'exclude-path-pattern:/**/*.css' \
  --set 'exclude-path-pattern:/**/*.eot' \
  --set 'exclude-path-pattern:/**/*.gif' \
  --set 'exclude-path-pattern:/**/*.ico' \
  --set 'exclude-path-pattern:/**/*.jpg' \
  --set 'exclude-path-pattern:/**/*.js' \
  --set 'exclude-path-pattern:/**/*.png' \
  --set 'exclude-path-pattern:/**/*.svg' \
  --set 'exclude-path-pattern:/**/*.ttf' \
  --set 'exclude-path-pattern:/**/*.woff' \
  --set 'exclude-path-pattern:/**/*.woff2' \
  --set 'exclude-path-pattern:/console/**' \
  --set 'exclude-path-pattern:/console/**/template/**' \
  --set log-file:logs/consent-trace \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy"
```

Correlating user and consent data

In some cases, the organization that has been granted consent by a group of users may need to perform an LDAP search so that they can act upon consent data in the aggregate. For example, a marketing group has collected consent to send a newsletter by email. A search must be performed that will list all of the consent records where the consent definition is `email` and the status is `accepted`. Those records must be correlated to user entries, and each user's email address must be retrieved.

This task is performed with an LDAP search on the PingDirectory Server. Every consent record has a `subject`, the user whose data is collected and stored. The Consent Service can be configured so that it stores the subject's DN in the `subjectDN` field.

In the LDAP schema:

- A consent record's `subjectDN` field is the `ping-consent-subject-dn` attribute.
- A consent record's status is the `ping-consent-state` attribute.
- A consent record's definition ID is in the `ping-consent-definition.id` JSON attribute field.
- And a user entry's email address is in the `mail` attribute.

The search will need to find all of the consent record entries where `ping-consent-definition.id` is `email` and the `ping-consent-status` is `accepted`. It then needs to correlate those consent record entries to user entries using `ping-consent-subject-dn`, and retrieve each user entry's `mail` attribute value. For example:

```
$ bin/ldapsearch \
   --baseDN "ou=consents,dc=example,dc=com" \
   --searchScope sub \
   --joinRule "dn:ping-consent-subject-dn" \
   --joinBaseDN "ou=people,dc=example,dc=com" \
   --joinScope sub \
   --joinRequestedAttribute mail
   '&(ping-consent-
definition:jsonObjectFilterExtensibleMatch:={ "filterType" : "equals",
 "field" : "id", "value" : "email" })(ping-consent-state=accepted)' \
   1.1

   # Join Result Control:
   #      OID:  1.3.6.1.4.1.30221.2.5.9
   #      Join Result Code:  0 (success)
   #      Joined With Entry:
   #           dn: uid=user.0,ou=People,dc=example,dc=com
   #           mail: user.0@example.com
   dn: entryUUID=9e481010-8330-425a-
bbf1-6637de053d48,ou=Consents,dc=example,dc=com

   # Result Code:  0 (success)
   # Number of Entries Returned:  1
```

The output listed under "Join Result Control" specifies the mail value.

Troubleshooting

The following are general guidelines for troubleshooting the Consent Service and any connection issues. When evaluating the configuration, make sure these issues are addressed first:

- Is the Consent Service enabled?
- Does the Consent Service base DN exist?
- Does the Consent Service's service account have the correct permissions?
- If the Consent Service should accept bearer tokens:
  - Are one or more Access Token Validators correctly configured?
  - Are the identity mappers for the Access Token Validators configured correctly?
  - Are the authorization servers correctly configured to issue tokens that the Consent Service will accept? Check the `audience`, `privileged-consent-scope`, and `unprivileged-consent-scope` properties of the Consent Service configuration.

- If privileged users are defined, are the members of the LDAP group specified by the Consent Service configuration's `privileged-users-group-dn` property?
- If there are applications that allow individuals to manage their own consents, is the system properly configured to map `actor` and `subject` DNs? Check the Consent Service configuration's `consent-record-identity-mapper` property.

Error cases

Consent Service is unavailable

If the Consent Service is unavailable, check that the service is enabled and that the communcation with the service is available. Confirm that the service account for the Consent Service has been properly provisioned. If the Consent Service resides on a PingDirectoryProxy Server, make sure that the service account exists on the PingDirectoryProxy Server and all PingDirectory Server behind the PingDirectoryProxy Server.

Requester lacks sufficient rights to perform operation

A request may be rejected with a 403 for the following reasons:

- The bearer token does not contain a required scope. Check the `privileged-consent-scope` and `unprivileged-consent-scope` properties of the Consent Service configuration.
- The bearer token does not contain a required `audience` claim. Check the `audience` property of the Consent Service configuration.
- Authentication was successful, but the requester is unprivileged and attempted to perform an operation that only a privileged requester may perform. For example, it may have attempted to act upon a consent record that it does not own, or it may have attempted to delete a consent record.

When using basic authentication, the requester must be listed in the Consent Service configuration `service-account-dn` property to be considered privileged.

Subject and actor do not match

Only a privileged requester can create or modify a consent record whose `subject` and `actor` values do not match.

Unindexed search

The Consent Service will not allow a client to make an unindexed search. In most cases, a client should be able to fix this by refining the search. For example, if a search by `subject` would be unindexed, perform a search by `subject` `definition` ID.

Search size limit exceeded

The Consent Service caps the maximum number of records that can be returned in a search result using its `search-size-limit` configuration property. This limit can be increased, or the client may be able to refine the search to produce fewer results.

Index

PingDirectory™

Release 7.0

Delegated Admin Application Guide

Notice

PingDirectory™ Product Documentation

Trademarks

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Contents

Chapter 1

Introduction to the Delegated Admin application

Topics:

Simple account administration can be performed by a group of administrators through the Delegated Admin application. This application manages identity attributes stored in the Directory Server.

Delegated Admin overview

Many organizations need to responsibly delegate some of the responsibility of managing identities in the Directory Server to reduce some of the burden of recurring tasks, such as password resets or updating account data. These simple administrative tasks can be delegated to a subset of administrators.

The Directory Server can be installed with the Delegated Admin application, which enables delegated administration of identities for scenarios such as help desk or customer service representatives (CSR) initiating a password reset and unlock; an HR administrator updating an employee profile; or an application administrator updating identity attributes.

Prerequisites

The Delegated Admin application manages user identity data stored in one or more Directory Servers. After a Directory Server instance is installed, the Delegated Admin application can be installed. You will need the HTTPS port that was configured during Directory Server setup.

A PingFederate server is used as the identity provider for authentication and authorization. Follow the PingFederate product documentation for installation and configuration instructions. Additional information is provided here for configuration with the Directory Server and the Delegated Admin application.

Chapter 2

Delegated Admin installation and configuration

Topics:

The Delegated Admin application relies on a previously installed PingDirectory Server and PingFederate Server. This section includes the configuration needed to support the Delegated Admin application.

Configuration overview

The Delegated Admin application must have a PingDirectory Server, and PingFederate Server installed. See the documentation for each product for installation instructions.

Configuration to support the Delegated Admin application on the PingDirectory Server includes:

- Configure administrators as Delegated Admin users.
- Configure attributes and attribute searching.

Configuration on the PingFederate Server includes:

- Configure PingFederate as the identity provider for the Delegated Admin application.
- Configure PingFederate as the OAuth server for the Delegated Admin application.
- Register the Delegated Admin application as a client.
- Register the PingDirectory Server as an OAuth token validator.

Authentication

The delegated administrator logs in to the Delegated Admin application through the PingFederate Server, which is configured as the authentication server and OpenID Connect (OIDC) provider. PingFederate validates the user credentials against the PingDirectory Server, and then issues an access token. The application obtains an access token from the PingFederate server, which encapsulates information `"claims"` about the user identity. The Delegated Admin application then presents this token to the Directory Server in the HTTP Authorization request header.

Interaction with the PingDirectory Server

The PingDirectory Server is configured to accept access tokens using Access Token Validators. The values that the PingFederate Server sets for access token `subject` claims must be mappable to a DN in the PingDirectory Server. Setting up an access token validator for use with the Delegated Admin application requires some coordination with the server configuration. In the suggested default configuration, the access token contains the entryUUID of the administrator user entry in the `sub` field. This value is mapped back to a PingDirectory Server entry using an Exact Match Identity Mapper.

Authorization by PingDirectory Server

Once validated, the PingDirectory Server checks the Delegated Admin configuration for authorization of the delegated administrator. Users or groups of users are authorized as delegated administrators in the Directory Server Administration Console, or with the `dsconfig` tool.

Client configuration

The Delegated Admin application uses PingFederate to authenticate users within the PingDirectory Server. Before installing and configuring the Delegated Admin application, you must configure two OAuth clients within PingFederate. One client is the Delegated Admin application, which will obtain an OIDC token that describes the authenticated user. The second client is the PingDirectory Server itself, which will call PingFederate to validate the OIDC token passed from the Delegated Admin application.

Specifically, the Delegated Admin OAuth client must be configured so that:

- The client id is "dadmin" and it requires no client secret key.
- The redirect URL is `https://SERVER:PORT/delegator/*`, where `SERVER:PORT` are the public host and port to access the Delegated Admin installation.
- The grant type is Implicit.
- The OIDC policy uses JWT tokens, where the entryUUID of the user is passed through the `subject` claim of the OIDC token.

The PingDirectory Server OAuth client must be configured so that:

- The client ID is "pingdirectory," and it does require a secret key.
- The grant type is Access Token Validation.

A detailed example of how PingFederate can be configured is available at the end of this guide.

Setup in a replicated PingDirectory Server environment

Running the Delegated Admin setup script requires special consideration in an environment that includes replicated PingDirectory Servers. If possible, setup the application after replication is enabled for the PingDirectory Servers. See the *PingDirectory Server Administration Guide* for details about server replication.

Set up Delegated Admin in an existing replicated topology

Complete the following steps if replication is already enabled for PingDirectory Servers.

1. If needed, configure the PingDirectory Servers to use a configuration group called "all-servers." This will ensure that configuration changes are applied to all servers in a topology.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:all-servers
```

2. Run the Delegated Admin setup script. See *Install the Delegated Admin application*.

Add an additional server to a replicated topology

To add an additional server to a topology containing a server already configured with the Delegated Admin application, perform the following the steps. In this example, "DS1" is the original PingDirectory Server, and "DS2" is the second server that will be added as a replica.

1. Run the `config-diff` command without arguments on DS1 to produce a batch file that contains configuration changes that will be applied to DS2.

```
$ bin/config-diff > config-changes.dsconfig
```

2. Apply the `config-changes.dsconfig` file to DS2.

```
$ bin/dsconfig --no-prompt \
  --batch-file config-changes.dsconfig \
  --applyChangeTo single-server
```

3. Restart DS2.
4. Enable replication between the two servers.

Install the Delegated Admin application

After the PingDirectory Server and the PingFederate Server have been configured, install the Delegated Admin application.

For systems such as Microsoft Windows that do not support bash scripts, copy the `delegated-admin-template.dsconfig` file and replace variables (`${variable}`) with actual values.

1. Download and unzip the Delegated Admin application package to a location on the PingDirectory Server.
2. Copy the Delegated Admin application folder and its contents into the PingDirectory Server's webapps directory.
3. In the Delegated Admin application directory, copy or rename the `example.config.js` file to `config.js`.

4. In an editor, open the `config.js` and change the values for the variables according to the local setup. The file contains comments and placeholders for necessary information. Save the changes.

   The client ID needed in this file is the one defined for the PingFederate configuration. In this file, it is the client intended for token issuance, such as `dadmin`.
5. In the PingDirectory Server root directory, run the following script to generate a `dsconfig` batch file:

   ```
   $ sh /webapps/delegator/set-up-delegator.sh
   ```

   The client ID and secret required for this script is the client configured for token verification on the PingFederate server, such as `pingdirectory`.
6. Apply the commands in the generated batch file to the PingDirectory Server. If installing in a replicated topology, add the `--applyChangeTo server-group` argument.

   ```
   $ ./bin/dsconfig \
     --bindDN "cn=Directory Manager" \
     --no-prompt \
     --batch-file webapps/delegator/delegated-admin.dsconfig -w
    directoryPassword
   ```
7. Go to `https://directoryHost:httpPort/delegator` to log in to the application.

Configure delegated administrators on the PingDirectory Server

To use the Delegated Admin application, an administrator must have more than valid credentials and an access token successfully validated by the PingDirectory Server. The administrator must be designated through the Directory Server configuration. To delegate users or groups as administrators, use the PingDirectory Server Administrative Console (Delegated Admin Resource Types), or the `dsconfig create-delegated-administrator` command.

The following sample commands illustrate configuration options for delegated administration, and are performed on the PingDirectory Server.

- The following command restricts an administrator to manage specified subtrees:

  ```
  $ bin/dsconfig create-delegated-administrator \
    --type-name users \
    --administrator-name admin1 \
    --set "admin-user-dn:uid=admin1,ou=people,dc=example,dc=com" \
    --set admin-scope:manages-specific-subtrees \
    --set “managed-subtree:ou=org1,dc=example,dc=com” \
    --set enabled:true
  ```
- An administrator could be restricted to manage the member users of one or more specified groups. The following example assumes the existence of a static or dynamic group entry whose members include the users to be managed.

  ```
  $ bin/dsconfig create-delegated-administrator \
    --type-name users \
    --administrator-name admin1 \
    --set "admin-user-dn:uid=admin1,ou=people,dc=example,dc=com" \
    --set admin-scope:manages-users-in-specific-groups \
    --set “managed-users-in-group:cn=User Group,dc=example,dc=com” \
    --set enabled:true
  ```
- Rather than delegate a single user as an administrator, it may be more convenient to delegate an entire group of users as administrators:

  ```
  $ bin/dsconfig create-delegated-administrator \
    --type-name users \
    --administrator-name admin-group1 \
    --set "admin-group-dn:cn=Admin Group,ou=people,dc=example,dc=com" \
  ```

```
  --set admin-scope:manages-all-entries \
  --set enabled:true
```

In this example, groups can be configured to use manage specific subtrees or groups with the `manages-specific-subtrees` or `manages-users-in-specific-groups` settings for the `admin-scope`. For more information about PingDirectory Server administrators and configuring dynamic and static groups, see the *PingDirectory Server Administration Guide*.

Configure attributes and attribute search on PingDirectory Server

The file used to install the Delegated Admin application specifies the object class of user entries through `structural-ldap-objectclass:inetOrgPerson`, and specifies a number of user attributes to be exposed by the application.

1. If needed, the attribute designated as the primary display attribute can be changed.

```
$ bin/dsconfig set-delegated-admin-resource-type-prop \
  --type-name users \
  --set primary-display-attribute-type:mail
```

2. Configure any additional user attributes to display in the Delegated Admin application by specifying the LDAP attribute type to be exposed and providing a display name for it.

```
$ bin/dsconfig create-delegated-admin-attribute \
  --type-name users \
  --attribute-type customAttr
  --set "display-name:My custom attribute"
```

3. When search text is entered in the application, the attributes to be searched in the PingDirectory Server are specified by the `search-filter-pattern` property. The PingDirectory Server must have appropriate attribute indices defined to satisfy the query. Use the following command to set the search filter.

```
$ bin/dsconfig set-delegated-admin-resource-type-prop \
  --type-name users \
  --set 'search-filter-pattern:(|(cn=*%%*)(mail=%%*)(uid=%%*))'
```

Appendix A

Sample PingFederate configuration

Topics:

PingFederate offers many configuration options. This section provides a sample configuration that can be used to support the Delegated Admin application.

PingFederate sample configuration

The following is a sample PingFederate configuration. Minimal support for the Delegated Admin applications includes configuring PingFederate to use the HTML Form Adapter to authenticate users for the Delegated Admin application, and configuring PingFederate to identify an authenticated user through the user's entryUUID, which is mapped to the `subject` of the OIDC token.

Configure PingFederate as the Identity Provider

This procedure configures the PingFederate Server as the identity provider for PingDirectory Server.

Note: Before starting, download the LDAPS certificate from PingDirectory Server. All other steps are performed on the PingFederate server. See the *PingDirectory Server Administration Guide* for details.

1. Add roles for OAuth and OIDC and Identity Provider under **Server Configuration** > **Server Settings** > **Roles and Protocols**.
2. Upload the PingDirectory Server LDAPS certificate in **Server Configuration** > **Trusted CAs**.
3. Add an LDAP data store in **Server Configuration** > **Data Stores**. Specify:

   a) The PingDirectory Server hostname and LDAPS port.

   b) Select **Use LDAPS**.

   c) Under **Advanced**, clear the **Verify LDAPS hostname** option.
4. Create the HTML form **IdP Adapter and Password Credential Validator** that is used to authenticate users against PingDirectory Server:

   a) Select **Identity Provider** > **Adapters** > **Create New Instance**.

   b) Select the HTML Form type.

   c) Scroll to bottom, and click **Manage Password Credential Validators**.

   d) Select **Create New Instance**.

   e) Select the LDAP Username Password Credential Validator.

   f) To use either email address or username to log in, enter the following search filter:

   ```
   (|(uid=${username})(mail=${username}))
   ```

   g) Extend the contract with "entryUUID" and "cn". These are used later.

   h) Click **Next** and **Save** until the Create Adapter Instance screen.

   i) Add a new row to Password Credential Validators, and choose the new LDAP Password Credential Validator.

   j) Extend the contract with "entryUUID" and "cn". These are used later.

   k) Check "entryUUID" for pseudonym, and **Save**.
5. Enable session tracking in **Identity Provider** > **Sessions**, and select the **Track adapter session for logout** and **Enable authentication sessions for all sources** options.
6. Click **Save**.

Configure the OAuth server

1. Click **IdP Adapter Mapping**, and add the new IdP adapter for creating OAuth grants. An additional attribute source is not needed. Fulfill the contract with the USER_KEY from Adapter "entryUUID" and USER_NAME from Adapter "cn", and click **Save**.
2. Click **Access Token Management** > **Create New Instance**. In this step, JSON Web Tokens are configured as an example:

   a) Select **JSON Web Tokens**.

   b) Choose one-way encryption for JWT, which only requires a symmetric key (not a certificate and private key). This requires the client to validate the token by hitting the validation endpoint on the server.

   c) Add a row to symmetric keys and use 32 bytes or 64 chars of hex.

   d) Choose the JWS Algorithm HMAC using SHA-256.

   e) Choose your symmetric key for Active Symmetric Key ID, and click **Next**.

f) Check all options.

g) List at least one attribute to be defined in the access token. Add `sub`, and click **Save**.

3. Click **Access Token Mapping** and map the access token attributes from the persistent grant.

   a) Choose **Default Context** and the new Access Token Manager.

   b) Map the `sub` claim from the USER_KEY of the persistent grant, which is mapped from the entryUUID.

4. Click **OpenID Connect Policy Management > Add Policy**.

   a) Choose the previously created Access Token Manager.

   b) Delete all of the extended contract attributes, except `sub`. Other scopes defined, if configured.

   c) Click **Next** to reach Contract Fulfillment.

   d) Fulfill the OIDC contract `sub` with the Access Token attribute `sub`.

   e) If a default OIDC policy is not already defined, set this new policy as the default, and click **Save**.

5. Add scopes for PingDirectory Server APIs.

   a) Click **Scope Management > Exclusive Scopes**.

   b) Add a value and description for `urn:pingidentity:directory-delegated-admin`.

   c) Click **Save**.

Configure the Delegated Admin application and PingDirectory Server as OAuth clients

The following steps configure PingDirectory Server as the token validator, and the Delegated Admin application as a new client.

1. Click **Create new client**, and set the client ID and name as PingDirectory Server.

   a) Use client secret for authentication and generate a new secret. Copy the key.

   b) Check **Access Token Validation** under Allowed Grant Types, and click **Save**.

      When creating a PingFederate Access Token Validator in PingDirectory Server, use the pingdirectory client ID and secret. An Identity Mapper in will also be used in the PingDirectory Server that will take the `sub` claim and match it against the entryUUID attribute.

2. Click **Create new client**, and set the client ID and name as dadmin.

   a) Do not configure authentication.

   b) Define the redirect URI as `"https://${directoryServer:port}/delegator/*"`, with the host and port of the PingDirectory Server.

   c) Check **Bypass Authorization Approval**, and choose the Implicit Grant type.

   d) Check **Allow Exclusive Scopes** and check `urn:pingidentity:directory-delegated-admin`.

   e) Check **Implicit** grant type.

   f) Choose the OIDC policy previously created, and click **Save**.

   g) Add `"https://${directoryServer:port}"` to the CORS origins under **OAuth Server > Authorization Server Settings > Cross-Origin Settings**.

Index

PingDataSync™

Release 7.0

Server Administration Guide

Ping Identity®

Self Service Account Manager™ PingPingDataSync Server™ Product Documentation

Trademarks

Disclaimer

The information provided in these documents is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Support

https://support.pingidentity.com/

Table of Contents

-

-

Preface

The PingDataSync Server Administration Guide provides procedures to install and manage the PingDataSync Server in a multi-client environment.

Audience

This guide is intended for administrators who are responsible for installing and managing servers in an identity enterprise environment. Knowledge of the following is recommended:

- Identity platforms and LDAP concepts.
- General system administration and networking practices.
- Java VM optimization.
- Application performance monitoring.
- Statistics and metrics basics.

Related documentation

The following documents represent the rest of the PingData product set and may be referenced in this guide:

- *PingData Directory Server Reference (HTML)*
- *PingData Directory Server Administration Guide (PDF)*
- *PingData Security Guide (PDF)*
- *PingData PingDataSync Server Reference Guide (HTML)*
- *PingData PingDataSync Server Administration Guide*
- *PingData PingDirectory Proxy Server Reference (HTML)*
- *PingData PingDirectory Proxy Server Administration Guide (PDF)*
- *PingData PingDataGovernance Server Reference (HTML)*

- *PingData PingDataGovernance Server Administration Guide (PDF)*
- *PingData Security Guide (PDF)*
- *LDAP SDK (HTML)*
- *Server SDK (HTML)*

Chapter 1: Introduction

The PingDataSync Server is a high-capacity, high-reliability data synchronization and transfer pipe between source and destination topologies.

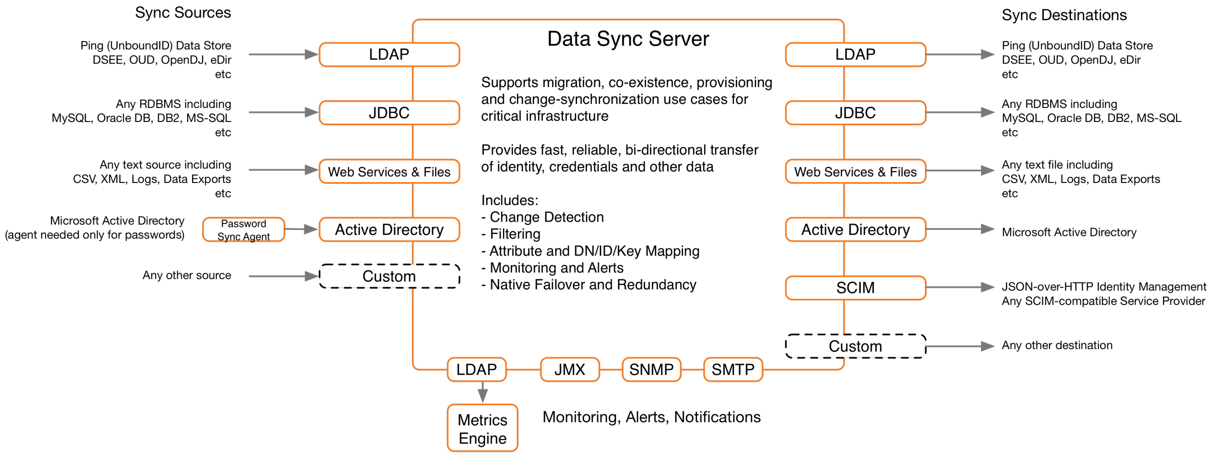
This chapter presents a general overview of the PingDataSync Server process and examples for use.

Topics include:

Overview of the PingDataSync Server

The PingDataSync Server is an efficient, Java-based server that provides high throughput, low-latency, and bidirectional real-time synchronization between two endpoint topologies consisting of Directory Servers, PingDirectory Proxy Servers, and/or Relational Database Management Systems (RDBMS) systems. The PingDataSync Server uses a dataless approach that synchronizes changes directly from the data sources in the background, so that applications can continue to update their data sources directly. The PingDataSync Server does not store any data from the endpoints themselves, thereby reducing hardware and administration costs. The server's high-availability mechanisms also makes it easy to fail over from the main PingDataSync Server to redundant instances.

The following is an overview of the components that may exist in a synchronization topology.


Sync Sources
Ping (UnboundID) Data Store DSEE, OUD, OpenDJ, eDir etc
Any RDBMS including MySQL, Oracle DB, DB2, MS-SQL etc
Any text source including CSV, XML, Logs, Data Exports etc
Microsoft Active Directory (agent needed only for passwords)
Password Sync Agent
Any other source
LDAP
JDBC
Web Services & Files
Active Directory
Custom
Data Sync Server
Supports migration, co-existence, provisioning and change-synchronization use cases for critical infrastructure
Provides fast, reliable, bi-directional transfer of identity, credentials and other data
Includes:
- Change Detection
- Filtering
- Attribute and DN/ID/Key Mapping
- Monitoring and Alerts
- Native Failover and Redundancy
LDAP
JDBC
Web Services & Files
Active Directory
SCIM
Custom
Sync Destinations
Ping (UnboundID) Data Store DSEE, OUD, OpenDJ, eDir etc
Any RDBMS including MySQL, Oracle DB, DB2, MS-SQL etc
Any text file including CSV, XML, Logs, Data Exports etc
Microsoft Active Directory
JSON-over-HTTP Identity Management Any SCIM-compatible Service Provider
Any other destination
LDAP
JMX
SNMP
SMTP
Metrics Engine
Monitoring, Alerts, Notifications

PingDataSync Server Topology Components

Designed to run with little administrative maintenance, the PingDataSync Server includes the following features:

- High performance and availability with built-in redundancy.
- Dataless virtual architecture for a small-memory footprint and easy maintenance.
- Hassle-free setup that enables mapping attribute names, values, and DNs between endpoints. For directory server endpoints, this enables making schema and Directory Information Tree (DIT) changes without custom coding and scripting.
- Multi-vendor directory server support including the Ping IdentityDirectory Server, Ping IdentityPingDirectory Proxy Server, Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server, Oracle/Sun Directory Server Enterprise Edition,

OpenDJ, and Microsoft Active Directory, and generic LDAP directories.

- RDBMS support including Oracle Database, and Microsoft SQL Server systems.
- Proxy Server support including the PingData PingDirectory Proxy Server and the Alcatel-Lucent 8661 Directory Proxy Server.
- Notification support that allows real-time change notifications to be pushed to client applications or services as they occur.

Data synchronization process

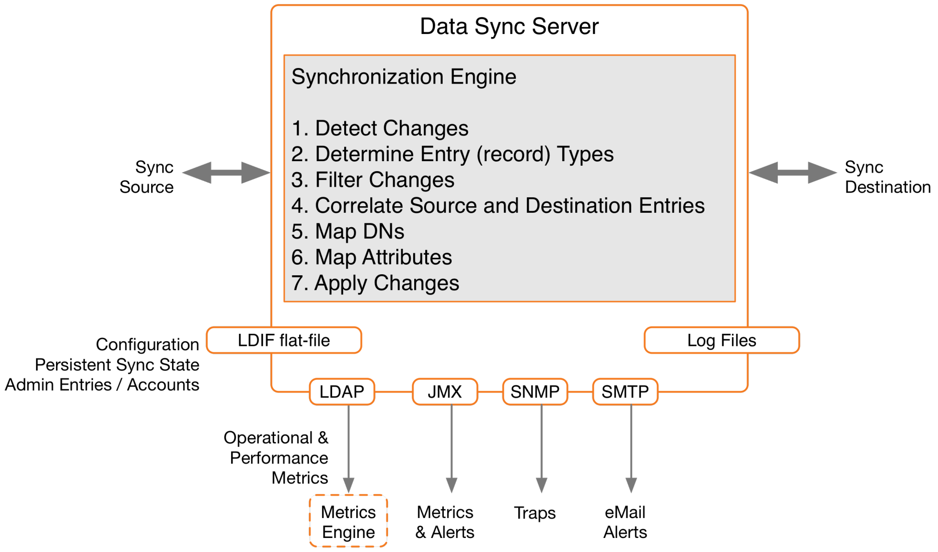
The PingDataSync Server performs point-to-point synchronization between a source endpoint and a destination endpoint. An endpoint is defined as any source or destination topology of directory or database servers.

The PingDataSync Server synchronizes data in one direction or bidirectionally between endpoints. For example, in a migration phase from Sun Directory Server to a PingData Directory Server, synchronization can occur in one direction from the source server to a staging server. With one-way synchronization, the source server is the authoritative endpoint for changes in the system. Bidirectional synchronization allows for parallel active installations between the source and the destination endpoints. With bidirectional synchronization, both endpoints are authoritative for the same set of attributes or for different sets of data.

ThePingDataSync Server also contains no single point of failure, either for detecting changes or for applying changes. PingDataSync Server instances themselves are redundant. There can be multiple instances running at a time, but only the server with the highest priority is actively synchronizing changes. The stand-by servers are constantly polling the active server instance to update their persistent state. This state contains the minimum amount of information needed to begin synchronization where the primary server left off, which logically is the last processed change number for the source server. In the case of a network partition, multiple servers can synchronize simultaneously without causing problems as they each verify the full entry before making changes.

Synchronization architecture

The PingDataSync Server uses a virtualized, dataless approach that does not store directory data locally. The log files, administrator entries, configuration, sync state information are stored as flat files (LDIF format) within the system. No additional database is required.



Synchronization Architecture

Change tracking, monitoring, and logging

The PingDataSync Server tracks and manages processes and server health with the following tools:

- **Change Tracking** – Each directory instance stores a separate entry under cn=changelog for every modification made to the directory. The PingDataSync Server provides full control over the synchronization process by determining which entries are synchronized, how they are correlated to the entries at the destination endpoint, and how they are transformed into the destination schema.
    - For the PingData Directory Server or Alcatel-Lucent 8661 Directory Server topologies, the PingDataSync Server uses the servers's LDAP Change Log for modification detection.
    - For Oracle/Sun Directory Server, OpenDJ, Oracle Unified Directory, and generic LDAP directory topologies, the PingDataSync Server uses the server's Retro Change Log, which provides a detailed summary of each change.
    - For Active Directory, the PingDataSync Server uses the DirSync control, which polls for object attribute changes.
    - For RDBMS systems, the PingDataSync Server uses an Ping Identity Server SDK plug-in to interface with a customized RDBMS change log table. Database triggers on each table record all INSERT, UPDATE, and DELETE operations to the change log table.

- **Monitoring, Alerts, and Alarms** – The PingDataSync Server supports several industry-standard, administrative protocols for monitoring, alarms, and alerts. System alarms and gauges can be configured to determine healthy performance thresholds and the server actions taken when performance values are outside the threshold. All administrative alarms are exposed over LDAP as entries under base DN `cn=alarms`. An administrative alert framework sends warnings, errors, or other server events through log messages, email, or JMX notifications. Administrative alerts are also exposed over LDAP as entries below base DN `cn=alerts`. Typical alert events are startup or shutdown, applied configuration changes, or synchronized resources unavailable.
- **Logging** – The PingDataSync Server provides standard logs (`sync`, `access`, `error`, `failed-operations`, `config-audit.log`, `debug`). The server can also be configured for multiple active sync logs. For example, each detected change, each dropped change, each applied change, or each failed change can be logged.

Synchronization Modes

The PingDataSync Server runs as a standalone Java process with two synchronization modes: standard and notification.

Standard Synchronization

In standard synchronization mode, thePingDataSync Server polls the directory server change log for create, modify, and delete operations on any entry. The server fetches the full entries from both the source and destination endpoints, and compares them to produce the minimal set of changes required to synchronize the destination with the source.

The following shows the standard synchronization change flow between two servers. The changes are processed in parallel, which increases throughput and offsets network latency.

Notification Synchronization

In notification synchronization mode, the PingDataSync Server skips the fetch and compare phases of processing and simply notifies the destination that a change has happened and provides the details of the change. Notification mode is currently available for the PingData and Alcatel-Lucent 8661 directory and proxy servers only.

PingDataSync Server Operations

The PingDataSync Server provides seamless integration between disparate systems to transform data using attribute and DN mappings. A bulk resynchronization operation can be run verify mappings and test synchronization settings.

Real-Time Synchronization

Real-time synchronization is performed with the `realtime-sync` utility. The `realtime-sync` utility polls the source server for changes and synchronizes the destination entries immediately. Once the server determines that a change should be synchronized, it fetches the full entry from the source. It then searches for the corresponding entry in the destination endpoint using correlation rules and applies the minimum set of changes to synchronize the attributes. The server fetches and compares the full entries to make sure it does not synchronize any old data from the change log.

After a synchronization topology is configured, run `resync` to synchronize the endpoints, and then run `realtime-sync` to start global synchronization.

The `realtime-sync` tool is used for the following tasks:

- Start or stop synchronization globally or for specific sync pipes only.
- Set a start point at which synchronization should begin such as the beginning or end of the change log, at a specified change number, at a specified change sequence number, or at a specified time frame in the change log.

Data Transformations

Data transformations alter the contents of synchronized entries between the source and destination directory server to handle variances in attribute names, attribute values, or DN structures. When entries are synchronized between a source and a destination server, the contents of these entries can be changed using attribute and DN mappings, so that neither server needs be aware of the transformations.

- **Attribute Mapping** – Any attribute in the entry can be renamed to fit the schema definitions from the source endpoint to the destination endpoint. This mapping makes it possible to synchronize information stored in one directory's attribute to an attribute with a different name in another directory server, or to construct an attribute using portions of the source attribute values.
- **DN Mapping** – Any DNs referenced in the entries can be transparently altered. This mapping makes it possible to synchronize data from a topology that uses one DIT structure to a system that uses a different DIT structure.

Bulk Resync

The `resync` tool performs a bulk comparison of entries on source topologies and destination topologies. The PingDataSync Server streams entries from the source, and either updates the corresponding destination entries or reports those that are different. The `resync` utility resides in the `/bin` folder (UNIX or LINUX) or `\bat` folder (Windows), and can be used for the following tasks:

- Verify that the two endpoints are synchronized after an initial configuration.
- Initially populate a newly configured target endpoint.

- Validate that the server is behaving as expected. The `resync` tool has a `--dry-run` option that validates that synchronization is operating properly, without updating any entries. This option also can be used to check attribute or DN mappings.
- Perform scheduled synchronization.
- Recover from a failover by resynchronizing entries that were modified since the last backup was taken.

The `resync` tool also enables control over what can be synchronized, such as:

- Include or exclude any source and destination attributes.
- Apply an LDAP filter to only sync entries created since that last time the tool ran.
- Synchronize only creations or only modifications.
- Change the logging verbosity.
- Set a limit on `resync` operations (such as 2000 operations per second) to reduce impact on endpoint servers.

The Sync Retry Mechanism

The PingDataSync Server is designed to quickly synchronize data and attempt a retry should an operation fail for any reason. The retry mechanism involves two possible retry levels, which are configurable on the Sync Pipe configuration using advanced Sync Pipe properties. For detailed information, see the *PingDataSync Server Reference Guide* for the Sync Pipe configuration parameters.

Retry involves two possible levels:

First Level Retry – If an operation fails to synchronize, the server will attempt a configurable number of retries. The total number of retry attempts is set in the `max-operation-attempts` property on the Sync Pipe. The property indicates how many times a worker thread should retry the operation before putting the operation into the second level of retry, or failing the operation altogether.

Second Level Retry – Once the `max-operation-attempts` property has been exceeded, the retry is sent to the second level, called the delayed-retry queue. The delayed-retry queue uses two advanced Sync Pipe properties to determine the number of times an operation should be retried in the background after a specified delay.

Operations that make it to this level will be retried after the `failed-op-background-retry-delay` property (default: 1 minute). Next, the PingDataSync Server checks the `max-failed-op-background-retries` property to determine the number of times a failed operation should be retried in the background. By default, this property is set to `0`, which indicates that no background retry should be attempted, and that the operation should be logged as failed.

> **Note**
> Background operations can hold up processing other changes, since the PingDataSync Server will only process up to the next 5000 changes while waiting for a retried operation to complete.

Retry can be controlled by the custom endpoint based on the type of error exception. When throwing an exception, the endpoint code can signal that a change should be aborted, retried a limited number of times, or retried an unlimited number of times. Some errors, such as endpoint server down, should be retried indefinitely.

If the `max-failed-op-background-retries` property has been exceeded, the retry is logged as a failure and appears in the `sync` and the `sync-failed-ops` logs.

Configuration Components

The PingDataSync Server supports the following configuration parameters that determine how synchronization takes place between directories or databases:

Sync Pipe – Defines a single synchronization path between the source and destination topologies. Every Sync Pipe has one or more Sync Classes that control how and what is synchronized. Multiple Sync Pipes can run in a single server instance.

Sync Source – Defines the directory topology that is the source of the data to be synchronized. A Sync Source can reference one or more supported external servers.

Sync Destination – Defines the topology of directory servers where changes detected at the Sync Source are applied. A Sync Destination can reference one or more external servers.

External Server – Defines a single server in a topology of identical, replicated servers to be synchronized. A single external server configuration object can be referenced by multiple Sync Sources and Sync Destinations.

Sync Class – Defines the operation types and attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated. A source entry is in one Sync Class and is determined by a base DN and LDAP filters. A Sync Class can reference zero or more Attribute Maps and DN Maps, respectively. Within a Sync Pipe, a Sync Class is defined for each type of entry that needs to be treated differently. For example, entries that define attribute mappings, or entries that should not be synchronized at all. A Sync Pipe must have at least one Sync Class but can refer to multiple Sync Class objects.

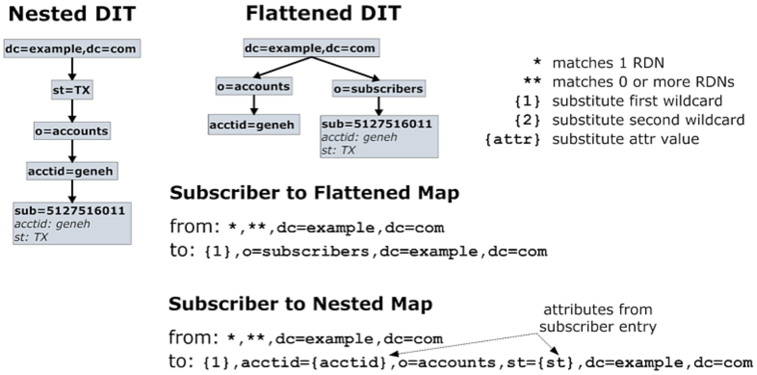
DN Map – Defines mappings for use when destination DNs differ from source DNs. These mappings allow the use of wild cards for DN transformations. A single wild card ("*") matches a single RDN component and can be used any number of times. The double wild card ("**") matches zero or more RDN components and can be used only once. The wild card values can be used in the `to-dn-pattern` attribute using `{1}` and their original index position in the pattern, or `{attr}` to match an attribute value. For example:

```
**,dc=myexample,dc=com->{1},o=example
```

Regular expressions and attributes from the user entry can also be used. For example, the following mapping constructs a value for the `uid` attribute, which is the RDN, out of the initials (first letter of `givenname` and `sn`) and the employee ID (`eid` attribute).

```
uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid},{2},o=example
```

The following illustrates a how a nested DIT can be mapped to a flattened structure.


Nested DIT
dc=example,dc=com
st=TX
o=accounts
acctid=geneh
sub=5127516011
acctid: geneh
st: TX
Flattened DIT
dc=example,dc=com
o=accounts
o=subscribers
acctid=geneh
sub=5127516011
acctid: geneh
st: TX
* matches 1 RDN
** matches 0 or more RDNs
{1} substitute first wildcard
{2} substitute second wildcard
{attr} substitute attr value
Subscriber to Flattened Map
from: *,**,dc=example,dc=com
to: {1},o=subscribers,dc=example,dc=com
Subscriber to Nested Map
attributes from subscriber entry
from: *,**,dc=example,dc=com
to: {1},acctid={acctid},o=accounts,st={st},dc=example,dc=com

Nested DIT Mapping

Attribute Map and Attribute Mappings – Defines a mapping for use when destination attributes differ from source attributes. A Sync Class can reference multiple attribute maps. Multiple Sync Classes can share the same attribute map. There are three types of attribute mappings:

- Direct Mapping – Attributes are directly mapped to another attribute: For example:

```
employeenumber->employeeid
```

- Constructed Mapping – Destination attribute values are derived from source attribute values and static text. For example:

```
{givenname}.{sn}@example.com->mail
```

- DN Mapping – Attributes are mapped for DN attributes. The same DN mappings that map entry DNs can be referenced. For example:

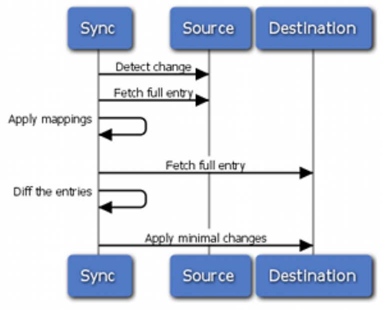
```
uid=jdoe,ou=People,dc=example,dc=com.
```

Sync Flow Examples

The PingDataSync Server processes changes by fetching the most up-to-date, full entries from both sides and then compares them. This process flow is called standard synchronization mode. The processing flow differs depending on the type of PingDataSync Serverchange (`ADD`, `MODIFY`, `DELETE`, `MODDN`) that is requested. The following examples show the control flow diagrams for the sync operations, especially for those cases when a `MODIFY` or a `DELETE` operation is dropped. The sync log records all completed and failed operations.

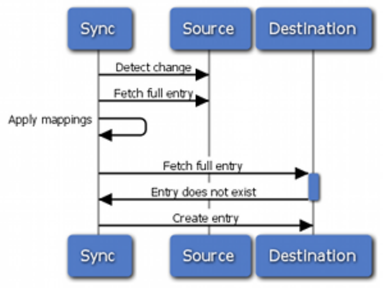
Modify Operation Example

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. Diff the computed destination entry and actual destination entry.
6. Apply the changes to the destination.



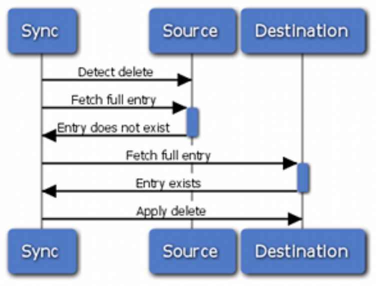
Add Operation Example

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. The entry or table row does not exist on the destination.
6. Create the entry or table row.



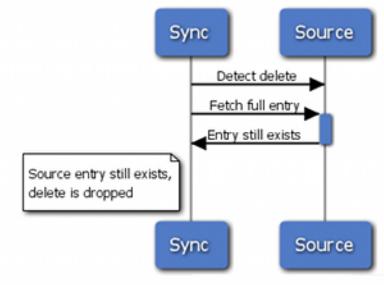
Delete Operation Example

1. Detect delete from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. The entry or table row exists on the destination.
6. Apply the delete on the destination.



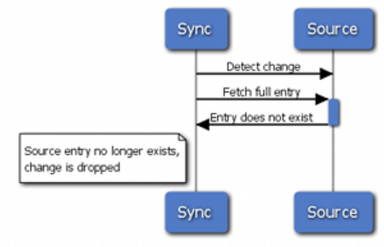
Delete After Source Entry is Re-Added

1. Detect delete from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. The entry or table row exists on the source.
4. Delete request is dropped.



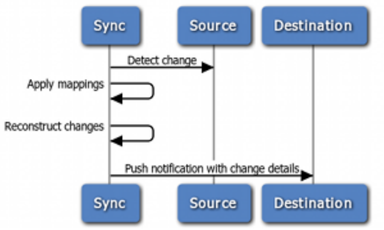
Standard Modify After Source Entry is Deleted

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. The entry does not exist.
4. Change request is dropped because the source entry no longer exists.



Notification Add, Modify, ModifyDN, and Delete

1. Detect change from the change log table on the source.
2. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
3. Reconstruct changed entries.
4. Push notification with change details to the destination.



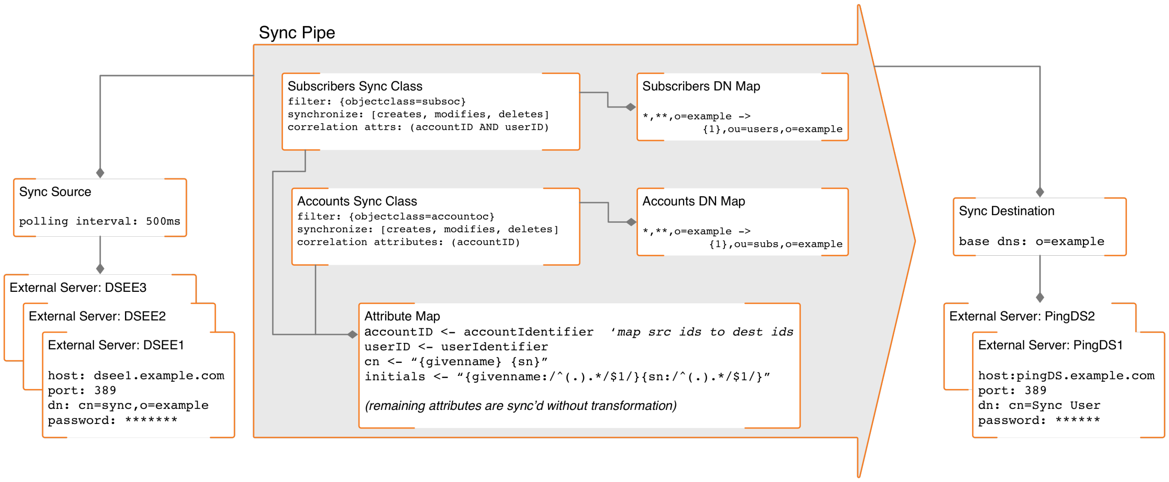
Sample Synchronization

The following is a synchronization migration example from a Sun Directory Server Enterprise Edition (DSEE) topology to the PingData Directory Server topology, including a change in the DIT structure to a flattened directory structure. The Sync Pipe connects the Sun Directory Server topology as the Sync Source and the Directory Server topology as the Sync Destination. Each endpoint is defined with three external servers in their respective topology. The Sync Pipe destination has its base DN set to `o=example`, which is used when performing LDAP searches for entries.

Two Sync Classes are defined: one for Subscribers, and one for Accounts. Each Sync Class uses a single "Sun DS to PingData Attribute Map" that has four attribute mappings defined. Each Sync Class also defines its own DN maps. For example, the Accounts Sync Class uses a DN map, called PingData Account Map, that is used to flatten a hierarchical DIT, so that the Account entries appear directly under `ou=accounts` as follows:

```
*,**,o=example -> {1},ou=accounts,o=example
```

With this mapping, if an entry DN has `uid=jsmith,ou=people,o=example`, then "`*`" matches `uid=jsmith`, "`**`" matches `ou=people`, and `{1}` matches `uid=jsmith`. Therefore, `uid=jsmith,ou=people,o=example` gets mapped to `uid=jsmith,ou=accounts,o=example`. A similar map is configured for the Subscribers Sync Class.



A Sample Synchronization Topology Configuration

Chapter 2: Installing the PingDataSync Server

This section describes how to install and run the PingDataSync Server. It includes pre-installation requirements and considerations.

Topics include:

Supported Platforms

The PingDataSync Server is a pure Java application. It is intended to run within the Java Virtual Machine on any Java Standard Edition (SE) or Enterprise Edition (EE) certified platform. For the list of supported platforms and Java versions, access the Ping Identity Customer Support Center portal or contact a PingData authorized support provider.

> **Note**
> It is highly recommended that a Network Time Protocol (NTP) system be in place so that multi-server environments are synchronized and timestamps are accurate.

Install the JDK

The Java 64-bit JDK is required on the server. Even if Java is already installed, create a separate Java installation for use by the server to ensure that updates to the system- wide Java installation do not inadvertently impact the installation.

Optimize the Linux Operating System

Configure the Linux filesystem by making the following changes.

> **Note**
> The server explicitly overrides environment variables like `PATH`, `LD_LIBRARY_PATH`, and `LD_PRELOAD` to ensure that settings used to start the server do not inadvertently impact its behavior. If these variables must be edited, set values by editing the `set_environment_vars` function of the `lib/_script-util.sh` script. Stop and restart the server for the change to take effect.

Set the file descriptor limit

The server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. If needed, increase the file descriptor limit on the operating system with the following procedure.

> **Note**
> If the operating system relies on `systemd`, refer to the Linux operating system documentation for instructions on setting the file descriptor limit.

1. Display the current hard limit of the system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

   ```
   ulimit -aH
   ```

2. Edit the `/etc/sysctl.conf` file. If the `fs.file-max` property is defined in the file, make sure its value is set to at least 65535. If the line does not exist, add the following to the end of the file:

   ```
   fs.file-max = 65535
   ```

3. Edit the /etc/security/limits.conf file. If the file has lines that set the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before #End of file). Insert a tab between the columns.

   ```
   * soft nofile 65535
   * hard nofile 65535
   ```

4. Reboot the system, and then use the ulimit command to verify that the file descriptor limit is set to 65535 with the following command:

   ```
   ulimit -n
   ```

Once the operating system limit is set, the number of file descriptors that the server will use can be configured by either using a NUM_FILE_DESCRIPTORS environment variable, or by creating a config/num-file-descriptors file with a single line such as, NUM_FILE_DESCRIPTORS=12345. If these are not set, the default of 65535 is used. This is strictly optional if wanting to ensure that the server shuts down safely prior to reaching the file descriptor limit.

> **Note**
> For RedHat 7 or later, modify the 20-nproc.conf file to set both the open files and max user processes limits:

```
/etc/security/limits.d/20-nproc.conf

Add or edit the following lines if they do not already exist:
*     soft    nproc     65536
*     soft    nofile    65536
*     hard    nproc     65536
*     hard    nofile    65536
root  soft    nproc     unlimited
```

Set the filesystem flushes

Linux systems running the ext3 filesystem only flush data to disk every five seconds. If the server is on a Linux system, edit the mount options to include the following:

```
commit=1
```

This variable changes the flush frequency from five seconds to one. Also, set the flush frequency in the /etc/fstab file to make sure the configuration remains after reboot.

Install sysstat and pstack on Red Hat

The server troubleshooting tool collect-support-data relies on the iostat, mpstat, and pstack utilities to collect monitoring, performance statistics, and stack trace information on the server's processes. For Red Hat systems, make sure that these packages are installed, for example:

```
$ sudo yum install sysstat gdb dstat -y
```

Install the dstat utility

The `dstat` utility is used by the `collect-support-data` tool.

Disable filesystem swapping

Any performance tuning services, like `tuned`, should be be disabled. If performance tuning is required, `vm.swappiness` can be set by cloning the existing performance profile then adding `vm.swappiness = 0` to the new profile's `tuned.conf` file in `/usr/lib/tuned/profile-name/tuned.conf`. The updated profile is then selected by running `tuned-adm profile customized_profile`.

Manage system entropy

Entropy is used to calculate random data that is used by the system in cryptographic operations. Some environments with low entropy may have intermittent performance issues with SSL-based communication. This is more typical on virtual machines, but can occur in physical instances as well. Monitor the `kernel.random.entropy_avail` in `sysctl` value for best results.

If necessary, update `$JAVA_HOME/jre/lib/security/java.security` to use `file:/dev/./urandom` for the `securerandom.source` property.

Enable the server to listen on privileged ports

Linux provides 'capabilities' used to grant specific commands the ability to do things that are normally only allowed for a root account. Instead of granting the ability to a specific user, capabilities are granted to a specific command. It may be convenient to enable the server to listen on privileged ports while running as a non-root user.

The `setcap` command is used to assign capabilities to an application. The `cap_net_bind_service` capability enables a service to bind a socket to privileged ports (port numbers less than 1024). If Java is installed in `/ds/java` (and the Java command to run the server is `/ds/java/bin/java`), the Java binary can be granted the `cap_net_bind_service` capability with the following command:

```
$ sudo setcap cap_net_bind_service=+eip /ds/java/bin/java
```

The java binary needs an additional shared library (`libjli.so`) as part of the Java installation. More strict limitations are imposed on where the operating system will look for shared libraries to load for commands that have capabilities assigned. So it is also necessary to tell the operating system where to look for this library. This can be done by creating the file `/etc/ld.so.conf.d/libjli.conf` with the path to the directory that contains the `libjli.so` file. For example, if the Java installation is in `/ds/java`, the contents of that file should be:

```
/ds/java/lib/amd64/jli
```

Run the following command for the change to take effect:

```
$ sudo ldconfig -v
```

Ping license keys

License keys are required to install all Ping products. Obtain licenses through Salesforce or from https://www.pingidentity.com/en/account/request-license-key.html.

- A license is always required for setting up a new single server instance and can be used site-wide for all servers in an environment. When cloning a server instance with a valid license, no new license is needed.
- A new license must be obtained when updating a server to a new major version, for example from 6.2 to 7.0. Licenses with no expiration date are valid until the server is upgraded to the next major version. A prompt for a new license is displayed during the update process.
- A license may expire on particular date. If a license does expire, obtain a new license and install it using `dsconfig` or the Administrative Console. The server will provide a notification as the expiration date approaches. License details are available using the server's `status` tool.

When installing the server, specify the license key file in one of the following ways:

- Copy the license key file to the server root directory before running setup. The interactive `setup` tool will discover the file and not require input. If the file is not in the server root, the setup tool will prompt for its location.
- If the license key is not in the server root directory, specify the `--licenseKeyFile` option for non-interactive setup, and the path to the file.

Install the PingDataSync Server

Use the `setup` tool to install the server. The server needs to be started and stopped by the user who installed it.

Note
A Windows installation requires that the Visual Studio 2010 runtime patch be installed prior to running the `setup` command.

1. Log in as a user, other than root.
2. Obtain the latest zip release bundle from Ping Identity and unpack it in a directory owned by this user.

```
$ unzip PingDataSync-<version>.zip
```

3. Change to the server root directory.

```
$ cd PingDataSync
```

4. Run the `setup` command.

```
$ ./setup
```

5. Type **yes** to accept the End-User License Agreement and press **Enter** to continue.
6. If adding this server to an existing PingDataSync Server topology, type **yes**, or press **Enter** to accept the default (no).
7. Enter the fully qualified host name or IP address of the local host.
8. Create the initial root user DN for the PingDataSync Server, or press **Enter** to accept the default (cn=Directory Manager).
9. Enter and confirm a password for this account.
10. Press **Enter** to enable server services and the Administrative Console.
11. Enter the port on which the PingDataSync Server will accept connections from HTTPS clients, or press **Enter** to accept the default.
12. Enter the port on which the PingDataSync Server will accept connections from LDAP clients, or press **Enter** to accept the default.
13. Press **Enter** to enable LDAPS, or enter **no**.
14. Press **Enter** to enable StartTLS, or enter **no**.
15. Select the certificate option for this server.
16. Choose the desired encryption for the directory data, backups, and log files from the choices provided:
    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.
    - Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
    - Do not encrypt server data.
17. Choose the option for the amount of memory that should be allocated to the server.
18. To start the server when the configuration is complete, press **Enter** for (yes).
19. A Setup Summary is displayed. choose the option to setup the server with the listed parameters, change the parameters, or cancel the setup.

After the server configuration is complete, the `create-sync-pipe-config` tool can be run to configure the synchronization environment.

The PingDataSync Server Administrative Console enables browser-based server management, the `dsconfig` tool enables command line management, and the Configuration API enables management by third-party interfaces.

Log into the Administrative Console

After the server is installed, access the Administrative Console, `https://<host>/console/login`, to verify the configuration and manage the server. The root user DN or the common name of a root user DN is required to log into the Administrative Console. For example, if the DN created when the server was installed is `cn=Directory Manager`, `directory manager` can be used to log into the Administrative Console.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package can be installed according to that container's documentation. Contact Ping Identity Customer Support for the package location and instructions.

Server folders and files

After the distribution file is unzipped, the following folders and command-line utilities are available:

Directories/Files/Tools	Description
ldif	Stores any LDIF files that you may have created or imported.
import-tmp	Stores temporary imported items.
classes	Stores any external classes for server extensions.
bak	Stores the physical backup files used with the backup command-line tool.
velocity	Stores Velocity templates that define the server's application pages.
update.bat, and update	The update tool for UNIX/Linux systems and Windows systems.
uninstall.bat, and uninstall	The uninstall tool for UNIX/Linux systems and Windows systems.
ping_logo.png	The image file for the Ping Identity logo.
setup.bat, and setup	The setup tool for UNIX/Linux systems and Windows systems.
revert-update.bat, and revert-update	The revert-update tool for UNIX/Linux systems and Windows systems.
README	README file that describes the steps to set up and start the server.
License.txt	Licensing agreement for the product.
legal-notices	Stores any legal notices for dependent software used with the product.
docs	Provides the release notes, Configuration Reference (HTML), API Reference, and all other product documentation.
metrics	Stores the metrics that can be gathered for this server and surfaced in the PingDataMetrics Server.

Directories/Files/Tools	Description
bin	Stores UNIX/Linux-based command-line tools.
bat	Stores Windows-based command-line tools.
lib	Stores any scripts, jar files, and library files needed for the server and its extensions.
collector	Used by the server to make monitored statistics available to the PingDataMetrics Server.
locks	Stores any lock files in the backends.
tmp	Stores temporary files.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
logs	Stores log files.
AD-Password-Sync-Agent.zip	The Active Directory Sync Agent package.

Start and stop the server

To start the PingDataSync Server, run the `bin/start-sync-server` command on UNIX or Linux systems (the `bat` folder on Microsoft Windows systems).

Start the Server as a Background Process

Navigate to the server root directory, and run the following command:

```
$ bin/start-server
```

For Windows systems:

```
$ bat/start-server
```

Start the server at boot time

By default, the server does not start automatically when the system is booted. To configure the server to start automatically, use the `create-rc-script` tool to create a run control script as follows:

1. Create the startup script. In this example `ds` is the user.

   ```
   $ bin/create-rc-script \
     --outputFile Ping-Identity-Sync.sh \
     --userName ds
   ```

2. Log in as root, move the generated `Ping-Identity-Sync.sh` script into the `/etc/init.d` directory, and create symlinks to it from the `/etc/rc3.d` (starting with an "S" to start

the server) and `/etc/rc0.d` directory (starting with a "K" to stop the server).

```
# mv Ping-Identity-Sync.sh /etc/init.d/
# ln -s /etc/init.d/Ping-Identity-Sync.sh /etc/rc3.d/S50-Ping-Identity-
Sync.sh
# ln -s /etc/init.d/Ping-Identity-Sync.sh /etc/rc0.d/K50-Ping-Identity-
Sync.sh
```

Stop the Server

If the PingDataSync Server has been configured to use a large amount of memory, it can take several seconds for the operating system to fully release the memory. Trying to start the server too quickly after shut down can fail because the system does not yet have enough free memory. On UNIX systems, run the `vmstat` command and watch the values in the "free" column increase until all memory held by the PingDataSync Server is released back to the system.

A configuration option can also be set that specifies the maximum shutdown time a process can take.

To stop the server, navigate to the server root directory and run the following command:

```
$ bin/stop-server
```

Restart the server

Restart the server using the `bin/stop-server` command with the `--restart` or `-R` option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again.

```
$ bin/stop-server --restart
```

Run the server as a Microsoft Windows service

The server can run as a Windows service on Windows Server 2012 R2 and Windows Server 2016. This enables log out of a machine without the server being stopped.

Register the service

Perform the following steps to register the server as a service:

1. Stop the server with `bin/stop-server`. A server cannot be registered while it is running.
2. Register the server as a service. From a Windows command prompt, run `bat/register-windows-service.bat`.
3. After a server is registered, start the server from the Windows Services Control Panel or with the `bat/start-server.bat` command.

Note
Command-line arguments for the `start-server.bat` and `stop-server.bat` scripts are not supported while the server is registered to run as a Windows service. Using a task to stop the server is also not supported.

Run multiple service instances

Only one instance of a particular service can run at one time. Services are distinguished by the `wrapper.name` property in the `<server-root>/config/wrapper-product.conf` file. To run additional service instances, change the `wrapper.name` property on each additional instance. Descriptions of the services can also be added or changed in the `wrapper-product.conf` file.

Deregister and uninstall

While a server is registered as a service, it cannot run as a non-service process or be uninstalled. Use the `bat/deregister-windows-service.bat` file to remove the service from the Windows registry. The server can then be uninstalled with the `uninstall.bat` script.

Log files

The log files are stored in `<server-root>/logs`, and filenames start with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or due to file size. Only the last three log files are retained. These configurations can be changed in the `<server-root>/config/wrapper.conf` file.

Uninstall the server

Use the `uninstall` command-line utility to uninstall the server using either interactive or non-interactive modes. Interactive mode provides options, progress, and a list of the files and directories that must be manually deleted if necessary.

Non-interactive mode, invoked with the `--no-prompt` option, suppresses progress information, except for fatal errors. All options for the `uninstall` command are listed with the `--help` option.

The `uninstall` command must be run as either the root user or the user account that installed the server.

Perform the following steps to uninstall in interactive mode:

1. Navigate to the server root directory.

   ```
   $ cd PingData<server>
   ```

2. Start the uninstall command:

   ```
   $ ./uninstall
   ```

3. Select the components to be removed, or press **Enter** to remove all components.
4. If the server is running, press **Enter** to shutdown the server before continuing.

5. Manually remove any remaining files or directories, if required.

Update servers in a topology

An update to the current release includes significant changes, and the introduction of a topology registry, which will store information previously stored in the admin backend (server instances, instance and secret keys, server groups, and administrator user accounts). For the admin backend to be migrated, the `update` tool must be provided LDAP authentication options to the peer servers of the server being updated.

The LDAP connection security options requested (either plain, TLS, StartTLS, or SASL) must be configured on every server in the topology. The LDAP credentials must be present on every server in the topology, and must have permissions to read from the admin backend and the config backend of every server in thetopology. For example, a root DN user with the `inherit-default-privileges` set to true (such as the `cn=Directory Manager` user) that exists on every server can be used.

After enabling or fixing the configuration of the LDAP connection handler(s) to support the desired connection security mechanism on each server, run the following `dsframework` command on the server being updated so that its admin backend has the most up-to-date information:

```
$ bin/dsframework set-server-properties \
  --serverID serverID \
  --set ldapport:port \
  --set ldapsport:port \
  --set startTLSEnabled:true
```

The `update` tool will verify that the following conditions are satisfied on every server in the topology before allowing the update:

- When the first server is being updated, all other servers in the topology must be online. When updating additional servers, all topology information will be obtained from one of the servers that has already been updated. The `update` tool will connect to the peer servers of the server being updated to obtain the necessary information to populate the topology registry. The provided LDAP credentials must have read permissions to the config and admin backends of the peer servers.
- The instance name is set on every server, and is unique across all servers in the topology. The instance name is a server’s identifier in the topology. After all servers in the topology have been updated, each server will be uniquely identified by its instance name. Once set, the name cannot be changed. If needed, the following command can be used to set the instance name of a server prior to the update:

  ```
  $ bin/dsconfig set-global-configuration-prop \
    --set instance-name:uniqueName
  ```

- The cluster-wide configuration is synchronized on all servers in the topology. Older versions have some topology configuration under `cn=cluster,cn=config` (JSON attribute and field constraints). These items did not support mirrored cluster-wide configuration data. An update should avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. To synchronize the cluster-wide configuration data across all servers in the topology, run the `config-diff` tool on each pair of servers to determine the differences, and use `dsconfig` to update each instance using the `config-diff` output. For example:

```
$ bin/config-diff --sourceHost hostName \
  --sourcePort port \
  --sourceBindDN bindDN \
  --sourceBindPassword password \
  --targetHost hostName \
  --targetPort port \
  --targetBindDN bindDN \
  --targetBindPassword password
```

If any of these conditions are not satisfied, the `update` tool will list all of the errors encountered for each server, and provide instructions on how to fix them.

Update the server

This procedure assumes that an existing version of the server is stored at `PingData-server-old`. Make sure a complete, readable backup of the existing system is available before upgrading the server. Also, make sure there is a clear backout plan and schedule.

1. Download the latest version of the server software and unzip the file. For this example, the new server is located in the `PingData-server-new` directory.
2. Use the `update` tool of the newly unzipped build to update the server. Make sure to specify the server instance that is being upgrading with the `--serverRoot` option. The server must be stopped for the update to be applied.

Reverting an Update

If necessary, a server can be reverted to the previous version using the `revert-update` tool. The tool accesses a log of file actions taken by the `update` tool to put the filesystem back to its prior state. If multiple updates have been run, the `revert-update` tool can be used multiple times to revert to each prior update sequentially. For example, the `revert-update` command can be run to revert to the server's previous state, then run again to return to its original state. The server is stopped during the revert-update process.

Note
Reverting an update is not supported for upgrades to version 7.0, due to the topology backend changes.

Use the `revert-update` tool in the server root directory revert back to the most recent version of the server:

```
$ PingData-server-old/revert-update
```

Install a failover server

Ping Identity supports redundant failover servers that automatically become active when the primary server is not available. Multiple servers can be present in the topology in a configurable prioritized order.

Before installing a failover server, have a primary server already installed and configured. When installing the redundant server, the installer will copy the first server's configuration.

The primary and secondary server configuration remain identical. Both servers should be registered to the `all servers` group and all `dsconfig` changes need to be applied to the server group `all servers`.

> **Note**
> If the primary server has extensions defined, they should also be installed on any cloned or redundant servers. If extensions are missing from a secondary server, the following message is displayed during the installation:
>
> ```
> Extension class <com.server.directory.sync.MissingSyncExtension> was not
> found. Run manage-extension --install to install your extensions. Re-run
> setup to continue.
> ```

To remove a failover server, use the `uninstall` command.

1. Unpack the Ping Identity server zip build. Name the unpacked directory something other than the first server instance directory.

   ```
   $ unzip PingData<server>-<version>.zip -d <server2>
   ```

2. Navigate to the server root directory.

3. Use the `setup` tool in interactive mode in Install the Server, or in non-interactive mode as follows:

   ```
   $ ./setup --localHostName <server2>.example.com --ldapPort 7389 \
     --masterHostName <server1>.example.com --masterPort 8389 \
     --masterUseNoSecurity \
     --acceptLicense \
     --rootUserPassword password \
     --no-prompt
   ```

   The secondary server is now ready to take over as a primary server in the event of a failover. No `realtime-sync` invocations are needed for this server.

4. Verify the configuration by using the `bin/status` tool. Each server instance is given a priority index. The server with the lowest priority index number has the highest priority.

   ```
   $ bin/status --bindPassword secret

   ...(status output)...
                                 --- Sync Topology ---
   Host:Port                              :Status       :Priority
   ---------------------------------------:-------------:---------------
   <server>.example.com:389 (this server) : Active      : 1
   <server>.example.com:389               : Unavailable : 2
   ```

5. Obtain the name of a particular server, run the `dsconfig` tool with the `list-external-servers` option.

   ```
   $ bin/dsconfig list-external-servers
   ```

6. To change the priority index of the server, use the `bin/dsconfig` tool:

   ```
   $ bin/dsconfig set-external-server-prop \
     --server-name <server2>.example.com:389 \
     --set <server>-priority-index:1
   ```

Administrative accounts

Users that authenticate to the Configuration API or the Administrative Console are stored in `cn=Root DNs,cn=config`. The `setup` tool automatically creates one administrative account when performing an installation. Accounts can be added or changed with the `dsconfig` tool.

Change the administrative password

Root users are governed by the Root Password Policy and by default, their passwords never expire. However, if a root user's password must be changed, use the `ldappasswordmodify` tool.

1. Open a text editor and create a text file containing the new password. In this example, name the file rootuser.txt.

   ```
   $ echo password > rootuser.txt
   ```

2. Use ldappasswordmodify to change the root user’s password.

   ```
   $ bin/ldappasswordmodify --port 1389 --bindDN "cn=Directory Manager" \
   --bindPassword secret --newPasswordFile rootuser.txt
   ```

3. Remove the text file.

   ```
   $ rm rootuser.txt
   ```

Chapter 3: Configuring the PingDataSync Server

The PingDataSync Server provides a suite of tools to configure a single server instance or server groups. All configuration changes to the server are recorded in the `config-audit.log`, and configuration is stored as a flat file (LDIF format) in the `cn=config` branch. Before configuring the PingDataSync Server, review Sync Configuration Components.

Topics include:

Configure failover behavior

Configure traffic through a load balancer

Configure authentication with a SASL external certificate

Server SDK extensions

Configuration checklist

Prior to any deployment, determine the configuration parameters necessary for the Synchronization topology. Gather the following:

External servers

External Server Type – Determine the type of external servers included in the synchronization topology. See Overview of the PingDataSync Server for a list of supported servers.

LDAP Connection Settings – Determine the host, port, bind DN, and bind password for each external server instance(s) included in the synchronization topology.

Security and Authentication Settings – Determine the secure connection types for each external server (SSL or StartTLS). Determine authentication methods for external servers such as simple, or external (SASL mechanisms). If synchronizing passwords, encoded or especially for clear-text, the connection should be secure. If synchronizing to or from a Microsoft Active Directory system, establish an SSL or StartTLS connection to the PingDataSync Server. Password encryption should also be enabled for synchronization from Active Directory, or when synchronizing clear-text passwords.

Sync Pipes

A Sync Pipe defines a single synchronization path between the source and destination targets. One Sync Pipe is needed for each point-to-point synchronization path defined for a topology.

Sync Source – Determine which external server is the Sync Source for the synchronization topology. A prioritized list of external servers can be defined for failover purposes.

Sync Destination – Determine which external server is the Sync Destination for the synchronization topology. A prioritized list of external servers can be defined for failover purposes.

Sync Classes

A Sync Class defines how attributes and DNs are mapped and how Source and Destination entries are correlated. For each Sync Pipe defined, define one or more Sync Classes with the following information:

Evaluation Order – If defining more than one Sync Class, the evaluation order of each Sync Class must be determined with the `evaluation-order-index` property. If there is an overlap between criteria used to identify a Sync Class, the Sync Class with the most specific criteria is used first.

Base DNs – Determine which base DNs contain entries needed in the Sync Class.

Include Filters – Determine the filters to be used to search for entries in the Sync Source.

Synchronized Entry Operations – Determine the types of operations that should be synchronized: creates, modifications, and/or deletes.

DNs – Determine the differences between the DNs from the Sync Source topology to the Sync Destination topology. Are there structural differences in each Directory Information Tree (DIT)? For example, does the Sync Source use a nested DIT and the Sync Destination use a flattened DIT?

Destination Correlation Attributes – Determine the correlation attributes that are used to associate a source entry to a destination entry during the synchronization process. During the configuration setup, one or more comma-separated lists of destination correlation attributes are defined and used to search for corresponding source entries. The PingDataSync Server maps all attributes in a detected change from source to destination attributes using the attribute maps defined in the Sync Class.

The correlation attributes are flexible enough so that several destination searches with different combinations of attributes can be performed until an entry matches. For LDAP server endpoints, use the distinguished name (DN) to correlate entries. For example, specify the attribute lists `dn,uid`, `uid,employeeNumber` and `cn,employeeNumber` to correlate entries in LDAP deployments. The PingDataSync Server will search for a corresponding entry that has the same `dn` and `uid` values. If the search fails, it then searches for `uid` and `employeeNumber`. Again if the search fails, it searches for `cn` and `employeeNumber`. If none of these searches are successful, the synchronization change would be aborted and a message logged.

To prevent incorrect matches, the most restrictive attribute lists (those that will never match the wrong entry) should be first in the list, followed by less restrictive attribute lists, which will be used when the earlier lists fail. For LDAP-to-LDAP deployments, use the DN with a combination of other unique identifiers in the entry to guarantee correlation. For other non-LDAP deployments, determine the attributes that can be synchronized across the network.

Attributes – Determine the differences between the attributes from the Sync Source to the Sync Destination, including the following:

- **Attribute Mappings** – How are attributes mapped from the Sync Source to the Sync Destination? Are they mapped directly, mapped based on attribute values, or mapped based on attributes that store DN values?
- **Automatically Mapped Source Attributes** – Are there attributes that can be automatically synchronized with the same name at the Sync Source to Sync Destination? For example, can direct mappings be set for `cn`, `uid`, `telephoneNumber`, or for all attributes?
- **Non-Auto Mapped Source Attributes** – Are there attributes that should not be automatically mapped? For example, the Sync Source may have an attribute, `employee`, while the Sync Destination may have a corresponding attribute, `employeeNumber`. If an attribute is not automatically mapped, a map must be provided if it is to be synchronized.
- **Conditional Value Mapping** – Should some mappings only be applied some of the time as a function of the source attributes? Conditional value mappings can be used with the

conditional-value-pattern property, which conditionalizes the attribute mapping based on the subtype of the entry, or on the value of the attribute. For example, this might apply if the adminName attribute on the destination should be a copy of the name attribute on the source, but only if the isAdmin attribute on the source is set to true. Conditional mappings are multi-valued. Each value is evaluated until one is matched (the condition is true). If none of the conditional mappings are matched, the ordinary mappings is used. If there is not an ordinary mapping, the attribute will not be created on the destination.

The Sync User account

The PingDataSync Server creates a Sync User account DN on each external server. The account (by default, cn=Sync User) is used exclusively by the PingDataSync Server to communicate with external servers. The entry is important in that it contains the credentials (DN and password) used by the PingDataSync Server to access the source and target servers. The Sync User account resides in different entries depending on the targeted system:

- For the Ping Identity Directory Server, Ping Identity PingDirectory Proxy Server, Alcatel-Lucent 8661 Directory Server, Alcatel-Lucent 8661 Directory Proxy Server, the Sync User account resides in the configuration entry (cn=Sync User, cn=Root DNs,cn=config).
- For Sun Directory Server, Sun DSEE, OpenDJ, Oracle Unified Directory, and generic LDAP directory topologies, the Sync User account resides under the base DN in the userRoot backend (cn=Sync User,dc=example,dc=com). The Sync User account should not reside in the cn=config branch for Sun Directory Server and DSEE machines.
- For Microsoft Active Directory servers, the Sync User account resides in the Users container (cn=Sync User,cn=Users,DC=adsync,DC=unboundid,DC=com).
- For Oracle and Microsoft SQL Servers, the Sync User account is a login account (SyncUser) with the sufficient privileges to access the tables to be synchronized.

In most cases, modifications to this account are rare. Make sure that the entry is not synchronized by setting up an optional Sync Class if the account resides in the userRoot backend (Sun Directory Server or Sun DSEE) or Users container (Microsoft Active Directory). For example, a Sync Class can be configured to have all CREATE, MODIFY, and DELETE operations set to false.

Configure the PingDataSync Server in Standard mode

The create-sync-pipe-config tool is used to configure Sync Pipes and Sync Classes. For bidirectional deployments, configure two Sync Pipes, one for each directional path.

Using the create-sync-pipe Tool to Configure Synchronization illustrates a bidirectional synchronization deployment in standard mode. The example assumes that two replicated topologies are configured:

- The first endpoint topology consists of two Sun LDAP servers: the main server and one failover. Both servers have Retro change logs enabled and contain the full DIT that will be synchronized to the second endpoint.
- The second endpoint topology consists of two Directory Servers: the main server and one failover. Both servers have change logs enabled and contain entries similar to the first endpoint servers, except that they use a `mail` attribute, instead of an `email` attribute.

A specific `mail` to `email` mapping must also be created to exclude the source attribute on the Sync Pipe going the other direction.

> **Note**
> If the source attribute is not excluded, the PingDataSync Server will attempt to create an `email` attribute on the second endpoint, which could fail if the attribute is not present in the destination server's schema.

Then, two Sync Classes are defined:

- One to handle the customized `email` to `mail` attribute mapping.
- Another to handle all other cases (the default Sync Class).

The `dsconfig` command is used to create the specific attribute mappings. The `resync` command is used to test the mappings. Synchronization can start using the `realtime-sync` command.

Use the create-sync-pipe tool to configure synchronization

Use the `create-sync-pipe-config` utility to configure a Sync Pipe. Once the configuration is completed, settings can be adjusted using the `dsconfig` tool.

> **Note**
> If servers have no base entries or data, the `cn=Sync User,cn=Root DNs,cn=config` account needed to communicate cannot be created. Make sure that base entries are created on the destination servers.

If synchronizing pre-encoded passwords to a Ping Directory Server destination, allow pre-encoded passwords in the default password policy. Password encryption must also be configured on the destination. Be sure that the password encryption algorithm is supported by both source and destination servers with the following command:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set allow-pre-encoded-passwords:true
```

Encrypted and clear-text passwords can be synchronized by configuring the Sync Destination `password-synchronization-format`, and `require-secure-connection-for-clear-text-passwords` properties.

> **Note**
> The `require-secure-connection-for-clear-text-passwords` property can be set to `false`

when working in a test environment. If the `password-synchronization-format` property is set to `clear-text`, and `require-secure-connection-for-clear-text-passwords` property is set to `true`, the connection must be secure. If a secure connection is not available, an error is generated and the password is not synchronized.

Perform the following steps to configure the PingDataSync Server using `create-sync-pipe-config`:

1. Start the PingDataSync Server.

   ```
   $ <server-root>/bin/start-server
   ```

2. From the `bin` directory, run the `create-sync-pipe-config` tool.

   ```
   $ bin/create-sync-pipe-config
   ```

3. On the Initial Synchronization Configuration Tool menu, press **Enter** (yes) to continue the configuration.
4. On the Synchronization Mode menu, press **Enter** to select Standard Mode.
5. On the Synchronization Directory menu, select **oneway (1)** or **bidirectional (2)** for the synchronization topology. This example assumes bidirectional synchronization.
6. On the Source Endpoint Type menu, select the directory or database server for the first endpoint.
7. On the Source Endpoint Name menu, type a name for the endpoint server, or press **Enter** accept the default.
8. On the Base DNs menu, type the base DN on the first endpoint topology where the entries will be searched. In this example, (`dc=example,dc=com`) is used.
9. Select an option for the server security.
10. Type the host name and listener port number for the source server, or accept the default. Make sure that the endpoint servers are online and running.
11. Enter another server host and port, or press **Enter** to continue.
12. Enter the Sync User account DN for the endpoint servers, or press **Enter** to accept the default (`cn=Sync User,cn=Root DNs,cn=config`).
13. Enter and confirm a password for this account.
14. The servers in the destination endpoint topology can now be configured. Repeat steps 6–13 to configure the second server.
15. Define the maximum age of changelog log entries, or press **Enter** to accept the default.
16. After the source and destination topologies are configured, the PingDataSync Server will "prepare" each external server by testing the connection to each server. This step determines if each account has the necessary privileges (root privileges are required) to communicate with and transfer data to each endpoint during synchronization.

17. Create a name for the Sync Pipe on the Sync Pipe Name menu, or press **Enter** to accept the default. Because this configuration is bidirectional, the following step is setting up a Sync Pipe path from the source endpoint to the destination endpoint. A later step will define another Sync Pipe from the Directory Server to another server.
18. On the Sync Class Definitions menu, type **Yes** to create a custom Sync Class. A Sync Class defines the operation types (creates, modifies, or deletes), attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated.
19. Enter a name for the new Sync Class, such as "server1_to_server2."
20. On the Base DNs for Sync Class menu, enter one or more base DNs to synchronize specific subtrees of a DIT. Entries outside of the specified base DNs are excluded from synchronization. Make sure the base DNs do not overlap.
21. On the Filters for Sync Class menu, define one or more LDAP search filters to restrict specific entries for synchronization, or press **Enter** to accept the default (no). Entries that do not match the filters will be excluded from synchronization.
22. On the Synchronized Attributes for Sync Class menu, specify which attributes will be automatically mapped from one system to another. This example will exclude the source attribute (`email`) from being auto-mapped to the target servers.
23. On the Operations for Sync Class menu, select the operations that will be synchronized for the Sync Class, or press **Enter** to accept the default (`1, 2, 3`).
24. Define a default Sync Class that specifies how the other entries are processed, or press **Enter** to create a Sync Class called "Default Sync Class."
25. On the Default Sync Class Operations menu, specify the operations that the default Sync Class will handle during synchronization, or press **Enter** to accept the default.
26. Define a Sync Pipe going from the Directory Server to the Sun Directory Server and exclude the `mail` attribute from being synchronized to the other endpoint servers.
27. Review the Sync Pipe Configuration Summary, and press **Enter** to accept the default (write configuration), which records the commands in a batch file (`<server-root>/sync-pipe-cfg.txt`). The batch file can be re-used to set up other topologies.

Apply the configuration changes to the local PingDataSync Server instance using a `dsconfig` batch file. Any Server SDK extensions, should be saved to the `<server-root>/lib/extensions` directory.

The next step will be to configure the attribute mappings using the `dsconfig` command.

Configuring attribute mapping

The following procedure defines an attribute map from the `email` attribute in the source servers to a `mail` attribute in the target servers. Both attributes must be valid in the target servers and must be present in their respective schemas.

> **Note**
> The following can also be done with `dsconfig` in interactive mode. Attribute mapping options are available from the PingDataSync Server main menu.

1. On the PingDataSync Server, run the `dsconfig` command to create an attribute map for the "SunDS>DS" Sync Class for the "Sun DS to Ping Identity DS" Sync Pipe, and then run the second `dsconfig` command to apply the new attribute map to the Sync Pipe and Sync Class.

```
$ bin/dsconfig --no-prompt create-attribute-map \
  --map-name "SunDS>DS Attr Map" \
  --set "description:Attribute Map for SunDS>Ping Identity Sync Class" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name "Sun DS to DS" \
  --class-name "SunDS>DS" \
  --set "attribute-map:SunDS>DS Attr Map" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. Create an attribute mapping (from `email` to `mail`) for the new attribute map.

```
$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name "SunDS>DS Attr Map" \
  --mapping-name mail --type direct \
  --set "description:Email>Mail Mapping" \
  --set from-attribute:email \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

3. For a bidirectional deployment, repeat steps 1–2 to create an attribute map for the DS>SunDS Sync Class for the Ping Identity DS to Sun DS Sync Pipe, and create an attribute mapping that maps `mail` to `email`.

```
$ bin/dsconfig --no-prompt create-attribute-map \
  --map-name "DS>SunDS Attr Map" \
  --set "description:Attribute Map for DS>SunDS Sync Class" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name "Ping Identity DS to Sun DS" \
  --class-name "DS>SunDS" \
  --set "attribute-map:DS>SunDS Attr Map" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name "DS>SunDS Attr Map" \
  --mapping-name email \
  --type direct \
  --set "description:Mail>Email Mapping" \
  --set from-attribute:mail \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configure server locations

The PingDataSync Server supports endpoint failover, which is configurable using the `location` property on the external servers. By default, the server prefers to connect to, and failover to, endpoints in the same location as itself. If there are no location settings configured, the PingDataSync Server will iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

> **Note**
> Location-based failover is only applicable for LDAP endpoint servers.

1. On the PingDataSync Server, run the `dsconfig` command to set the location for each external server in the Sync Source and Sync Destination. For example, the following command sets the location for six servers in two data centers, `austin` and `dallas`.

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:1389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:2389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:3389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:4389 \
  --set location:dallas
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:5389 \
  --set location:dallas
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:6389 \
  --set location:dallas
```

2. Run `dsconfig` to set the location on the Global Configuration. This is the location of the PingDataSync Server itself. In this example, set the location to "austin."

```
$ bin/dsconfig set-global-configuration-prop \
  --set location:austin
```

Use the Configuration API

PingData servers provide a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers should allow the `application/json` content type.

The server includes a servlet extension that provides read and write access to the server's configuration over HTTP. The extension is enabled by default for new installations, and can be enabled for existing deployments by simply adding the extension to one of the server's HTTP Connection Handlers, as follows:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add http-servlet-extension:Configuration
```

The API is made available on the HTTPS Connection handler's `host:port` in the `/config` context. Due to the potentially sensitive nature of the server's configuration, the HTTPS Connection Handler should be used, for hosting the Configuration extension.

Authentication and authorization

Clients must use HTTP Basic authentication to authenticate to the Configuration API. If the username value is not a DN, then it will be resolved to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper, or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. For example:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Configuration \
  --set "identity-mapper:Alternative Identity Mapper”
```

To access configuration information, users must have the appropriate privileges:

- To access the `cn=config` backend, users must have the `bypass-acl` privilege or be allowed access to the configuration using an ACI.
- To read configuration information, users must have the `config-read` privilege.
- To update the configuration, users must have the `config-write` privilege.

Relationship between the Configuration API and the dsconfig tool

The Configuration API is designed to mirror the `dsconfig` tool, using the same names for properties and object types. Property names are presented as hyphen case in `dsconfig` and as camel-case attributes in the API. In API requests that specify property names, case is not important. Therefore, baseDN is the same as baseDn. Object types are represented in hyphen case. API paths mirror what is in dsconfig. For example, the dsconfig list-connection-handlers command is analogous to the API's /config/connection-handlers path. Object types that appear in the schema URNs adhere to a type:subtype syntax. For example, a Local DB Backend's schema URN is urn:unboundid:schemas:configuration:2.0:backend:local-db. Like the dsconfig tool, all configuration updates made through the API are recorded in logs/config-audit.log.

The API includes the filter, sort, and pagination query parameters described by the SCIM specification. Specific attributes may be requested using the attributes query parameter, whose value must be a comma-delimited list of properties to be returned, for example attributes=baseDN,description. Likewise, attributes may be excluded from responses by specifying the excludedAttributes parameter. See Sorting and Filtering with the Configuration API for more information on query parameters.

Operations supported by the API are those typically found in REST APIs:

HTTP Method	Description	Related dsconfig Example
GET	Lists the attributes of an object when used with a path representing an object, such as /config/global-configuration or /config/backends/userRoot. Can also list objects when used with a path representing a parent relation, such as /config/backends.	get-backend-prop list-backends get-global-configuration-prop
POST	Creates a new instance of an object when used with a relation parent path, such as config/backends.	create-backend
PUT	Replaces the existing attributes of an object. A PUT operation is similar to a PATCH operation, except that the PATCH is determined by determining the difference between an existing target object and a supplied source object. Only those attributes in the source object are modified in the target object. The target object is specified using a path, such as /config/backends/userRoot.	set-backend-prop set-global-configuration-prop
PATCH	Updates the attributes of an existing object when used with a path representing an object, such as /config/backends/userRoot. See PATCH Example.	set-backend-prop set-global-configuration-prop
DELETE	Deletes an existing object when used with a path representing an object, such as /config/backends/userRoot.	delete-backend

The OPTIONS method can also be used to determine the operations permitted for a particular path.

Object names, such as userRoot in the Description column, must be URL-encoded in the path segment of a URL. For example, %20 must be used in place of spaces, and %25 is used in place of the percent (%) character. So the URL for accessing the HTTP Connection Handler object is:

```
/config/connection-handlers/http%20connection%20handler
```

GET Example

The following is a sample GET request for information about the `userRoot` backend:

```
GET /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
```

The response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://localhost:5033/config/backends/userRoot"
  },
  "backendID": "userRoot2",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example2,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "10",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "1 m",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "0",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "0",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode": "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "enabled": "false",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "false",
  "id2childrenIndexEntryLimit": "66",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "jeProperty": [
    "je.cleaner.adjustUtilization=false",
    "je.nodeMaxEntries=32"
  ],
  "numRecentChanges": "50000",
  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "5000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

GET list example

The following is a sample GET request for all local backends:

```
GET /config/backends
Host: example.com:5033
Accept: application/scim+json
```

The response (which has been shortened):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=admin data"
      ],
      "enabled": "true",
      "isPrivateBackend": "true",
      "javaClass": "com.unboundid.directory.server.backends.LDIFBackend",
      "ldifFile": "config/admin-backend.ldif",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "false",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
      ],
      "id": "ads-truststore",
      "meta": {
        "resourceType": "Trust Store Backend",
        "location": "http://localhost:5033/config/backends/ads-truststore"
      },
      "backendID": "ads-truststore",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=ads-truststore"
      ],
      "enabled": "true",
      "javaClass":
"com.unboundid.directory.server.backends.TrustStoreBackend",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "true",
      "trustStoreFile": "config/server.keystore",
      "trustStorePin": "********",
      "trustStoreType": "JKS",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:alarm"
      ],
      "id": "alarms",
      "meta": {
        "resourceType": "Alarm Backend",
        "location": "http://localhost:5033/config/backends/alarms"
      },
 ...
```

PATCH example

Configuration can be modified using the HTTP PATCH method. The PATCH request body is a JSON object formatted according to the SCIM patch request. The Configuration API, supports a subset of possible values for the `path` attribute, used to indicate the configuration attribute to modify.

The configuration object's attributes can be modified in the following ways. These operations are analogous to the `dsconfig modify-[object]` options.

- An operation to set the single-valued `description` attribute to a new value:

  ```
  {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --set "description:A new backend"
  ```

- An operation to add a new value to the multi-valued `jeProperty` attribute:

  ```
  {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --add je-property:je.env.backgroundReadLimit=0
  ```

- An operation to remove a value from a multi-valued property. In this case, `path` specifies a SCIM filter identifying the value to remove:

  ```
  {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --remove je-property:je.cleaner.adjustUtilization=false
  ```

- A second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify, and a SCIM filter whose attribute is `value`:

  ```
  {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --remove je-property:je.nodeMaxEntries=32
  ```

- An option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value:

  ```
  {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  }
  ```

  is analogous to:

  ```
  $ dsconfig set-backend-prop --backend-name userRoot \
    --reset id2childrenIndexEntryLimit
  ```

The following is the full example request. The API responds with the entire modified configuration object, which may include a SCIM extension attribute `urn:unboundid:schemas:configuration:messages` containing additional instructions:

Example request:

```
PATCH /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json

{
  "schemas" : [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
```

```
  } ]
}
```

Example response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot2",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot2"
  },
  "backendID": "userRoot2",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example2,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "10",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "1 m",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "0",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "0",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode": "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "123",
  "enabled": "false",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "false",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "jeProperty": [
  "\"je.env.backgroundReadLimit=0\""
  ],
  "numRecentChanges": "50000",
  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "5000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled",
  "urn:unboundid:schemas:configuration:messages:2.0": {
    "requiredActions": [
      {
        "property": "jeProperty",
        "type": "componentRestart",
        "synopsis": "In order for this modification to take effect,
         the component must be restarted, either by disabling and
         re-enabling it, or by restarting the server"
      },
      {
        "property": "id2childrenIndexEntryLimit",
        "type": "other",
        "synopsis": "If this limit is increased, then the contents
```

```
         of the backend must be exported to LDIF and re-imported to
         allow the new limit to be used for any id2children keys
         that had already hit the previous limit."
      }
    ]
  }
}
```

API paths

The Configuration API is available under the `/config` path. A full listing of root sub-paths can be obtained from the `/config/ResourceTypes` endpoint:

```
GET /config/ResourceTypes
Host: example.com:5033
Accept: application/scim+json
```

Sample response (abbreviated):

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "dsee-compat-access-control-handler",
      "name": "DSEE Compat Access Control Handler",
      "description": "The DSEE Compat Access Control
         Handler provides an implementation that uses syntax
         compatible with the Sun Java System Directory Server
         Enterprise Edition access control handler.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/dsee-compat-
access-control-handler"
      }
    },
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "access-control-handler",
      "name": "Access Control Handler",
      "description": "Access Control Handlers manage the
         application-wide access control. The server's access
         control handler is defined through an extensible
         interface, so that alternate implementations can be created.
         Only one access control handler may be active in the server
         at any given time.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/ResourceTypes/access-
control-handler"
      }
    },
    {
...
```

The response's `endpoint` elements enumerate all available sub-paths. The path `/config/access-control-handler` in the example can be used to get a list of existing access control handlers, and create new ones. A path containing an object name like `/config/backends/{backendName}`, where `{backendName}` corresponds to an existing backend (such as `userRoot`) can be used to obtain an object's properties, update the properties, or delete the object.

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted nor created. These paths can be differentiated from others by their singular, rather than plural, relation name (for example `global-configuration`).

Sorting and filtering configuration objects

The Configuration API supports SCIM parameters for filter, sorting, and pagination. Search operations can specify a SCIM filter used to narrow the number of elements returned. See the SCIM specification for the full set of operations for SCIM filters. Clients may also specify sort parameters, or paging parameters. As previously mentioned, clients may specify attributes to include or exclude in both get and list operations.

GET Parameters for Sorting and Filtering

GET Parameter	Description
filter	Values can be simple SCIM filters such as `id eq "userRoot"` or compound filters like `meta.resourceType eq "Local DB Backend"` and `baseDn co "dc=exmple,dc=com"`.
sortBy	Specifies a property value by which to sort.
sortOrder	Specifies either `ascending` or `descending` alphabetical order.
startIndex	1-based index of the first result to return.
count	Indicates the number of results per page.

Update properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH. With PUT, the server computes the differences between the object in the request with the current version in the server, and performs modifications where necessary. The server will never remove attributes that are not specified in the request. The API responds with the entire modified object.

Request:

```
PUT /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
{
  "description" : "A new description."
}
```

Response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode": "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "abc",
  "enabled": "true",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "true",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "numRecentChanges": "50000",
  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative actions

Updating a property may require an administrative action before the change can take effect. If so, the server will return `200 Success`, and any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

For example, changing the `jeProperty` of a backend will result in the following:

```
"urn:unboundid:schemas:configuration:messages:2.0": {
  "requiredActions": [
    {
      "property": "baseContextPath",
      "type": "componentRestart",
      "synopsis": "In order for this modification to
         take effect, the component must be restarted,
         either by disabling and re-enabling it, or by
         restarting the server"
    },
    {
      "property": "id2childrenIndexEntryLimit",
      "type": "other",
      "synopsis": "If this limit is increased, then the
         contents of the backend must be exported to LDIF
         and re-imported to allow the new limit to be used
         for any id2children keys that had already hit the
         previous limit."
    }
  ]
}
...
```

Update servers and server groups

Servers can be configured as part of a server group, so that configuration changes that are applied to a single server, are then applied to all servers in a group. When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query parameter. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `singleServer` or `serverGroup` can be specified. For example:

```
https://example.com:5033/config/Backends/userRoot?applyChangeTo=singleServer
```

Note

This does not apply to mirrored subtree objects, which include Topology and Cluster level objects. Changes made to mirrored objects are applied to all objects in the subtree.

Configuration API Responses

Clients of the API should examine the HTTP response code in order to determine the success or failure of a request. The following are response codes and their meanings:

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, or object properties, administrative actions.
204 No Content	The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation.	None.
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message.
401 Unauthorized	User authentication is required. Some user agents such as browsers may respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid.	None.
403 Forbidden	The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted.	None.
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message.
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists or an attempt was made to delete an object that is referred to by another object.	Error summary and optional message.
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None.
500 Server Error	The server encountered an unexpected error. Please report server errors to customer support.	Error summary and optional message.

An application that uses the Configuration API should limit dependencies on particular text appearing in error message content. These messages may change, and their presence may depend on server configuration. Use the HTTP return code and the context of the request to create a client error message. The following is an example encoded error message:

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "status": 404,
  "scimType": null,
  "detail": "The Local DB Index does not exist."
}
```

Configuration with the dsconfig tool

The Ping Identity servers provide several command-line tools for management and administration. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

The `dsconfig` tool is the text-based management tool used to configure the underlying server configuration. The tool has three operational modes:

- Interactive mode
- Non-interactive mode
- Batch mode

The `dsconfig` tool also offers an offline mode using the `--offline` option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. View detailed argument options and examples by typing `--help` with the command.

```
$ bin/dsconfig --help
```

To list the subcommands for each command:

```
$ bin/dsconfig --help-subcommands
```

To list more detailed subcommand information:

```
$ bin/dsconfig list-log-publishers --help
```

Use dsconfig in interactive mode

Running `dsconfig` in interactive command-line mode provides a menu-driven interface for accessing and configuring the PingData server. To start `dsconfig` in interactive mode, run the tool without any arguments:

```
$ bin/dsconfig
```

Running the tool requires server connection and authentication information. After connection information is confirmed, a menu of the available operation types is displayed.

Use dsconfig in non-interactive mode

Non-interactive command-line mode provides a simple way to make arbitrary changes to the server, and to use administrative scripts to automate configuration changes. To make changes to multiple configuration objects at the same time, use batch mode.

The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The `--no-prompt` argument specifies non-interactive mode. The `{globalArgs}` argument provides a set of arguments that specify how to connect and authenticate to the server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on the implementation. The `{subcommand}` specifies which general action to perform. The following uses standard LDAP connections:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

The following uses SASL GSSAPI (Kerberos) parameters:

```
$ bin/dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The `{subcommandArgs}` argument contains a set of arguments specific to the particular task. To always display the advanced properties, use the `--advanced` command-line option.

> **Note**
> Global arguments can appear anywhere on the command line. The subcommand-specific arguments can appear anywhere after the subcommand.

Use dsconfig batch mode

The `dsconfig` tool provides a batching mechanism that reads multiple invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The `--no-prompt` option is required with `dsconfig` in batch mode.

```
$ bin/dsconfig --no-prompt \
  --hostname host1 \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --batch-file /path/to/sync-pipe-config.txt
```

If a `dsconfig` command has a missing or incorrect argument, the command will fail and stop the batch process without applying any changes to the server. A `--batch-continue-on-error` option is available, which instructs `dsconfig` to apply all changes and skip any errors.

View the `logs/config-audit.log` file to review the configuration changes made to the server, and use them in the batch file. The batch file can have blank lines for spacing, and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The PingDataSync Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to Ping Identity PingDataSync Server machines.

Topology configuration

Topology configuration enables grouping servers and mirroring configuration changes automatically. It uses a master/slave architecture for mirroring shared data across the topology. All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group.

> **Note**
> To remove a server from the topology, it must be uninstalled with the `uninstall` tool.

Topology master requirements and selection

A topology master server receives any configuration change from other servers in the topology, verifies the change, then makes the change available to all connected servers when they poll the master. The master always sends a digest of its subtree contents on each update. If the node has a different digest than the master, it knows it's not synchronized. The servers will pull the entire subtree from the master if they detect that they are not synchronized. A server may detect it is not synchronized with the master under the following conditions:

- At the end of its periodic polling interval, if a server's subtree digest differs from that of its master, then it knows it's not synchronized.
- If one or more servers have been added to or removed from the topology, the servers will not synchronized.

The master of the topology is selected by prioritizing servers by minimum supported product version, most available, newest server version, earliest start time, and startup UUID (a smaller UUID is preferred).

After determining a master, the topology data is reviewed from all available servers (every five seconds by default) to determine if any new information makes a server better suited to being the master. If a new server can be the master, it will communicate that to the other servers, if no other server has advertised that it should be the master. This ensures that all servers accept the same master at approximately the same time (within a few milliseconds of each other). If there is no better master, the initial master maintains the role.

After the best master has been selected for the given interval, the following conditions are confirmed:

- A majority of servers is reachable from that master. (The master server itself is considered while determining this majority.)
- There is only a single master in the entire topology.

If either of these conditions is not met, the topology is without a master and the peer polling frequency is reduced to 100 milliseconds to find a new master as quickly as possible. If there is no master in the topology for more than one minute, a `mirrored-subtree-manager-no-master-found` alarm is raised. If one of the servers in the topology is forced as master with the force-as-master-for-mirrored-data option in the Global Configuration configuration object, a mirrored-subtree-manager-forced-as-master-warning warning alarm is raised. If multiple servers have been forced as masters, then a mirrored-subtree-manager-forced-as-master-error critical alarm will be raised.

Topology components

When a server is installed, it can be added to an existing topology, which will clone the server's . Topology settings are designed to operate without additional configuration. If required, some settings can be adjusted to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the Global Configuration and in the Config File Handler Backend. Though they are topology settings, they are unique to each server and are not mirrored. Settings must be kept the same on all servers.

The Global Configuration object contains a single topology setting, force-as-master-for-mirrored-data. This should be set to true on only one of the servers in the topology, and is used only if a situation occurs where the topology cannot determine a master because a majority of servers is not available. A server with this setting enabled will be assigned the role of master, if no suitable master can be determined. See Topology master requirements and selection for details about how a master is selected for a topology.

The Config File Handler Backend defines three topology (mirrored-subtree) settings:

- mirrored-subtree-peer-polling-interval – Specifies the frequency at which the server polls its topology peers to determine if there are any changes that may warrant a new master selection. A lower value will ensure a faster failover, but it will also cause more traffic among the peers. The default value is five seconds. If no suitable master is found, the polling frequency is adjusted to 100 milliseconds until a new master is selected.
- mirrored-subtree-entry-update-timeout – Specifies the maximum length of time to wait for an update operation (add, delete, modify or modify-dn) on an entry to be applied by the master on all of the servers in the topology. The default is 10 seconds. In reality, updates can take up to twice as much time as this timeout value if master selection is in progress at the time the update operation was received.
- mirrored-subtree-search-timeout – Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

Topology meta-data is stored under the cn=topology,cn=config subtree and cluster data is stored under the cn=cluster,cn=config subtree. The only setting that can be changed is the cluster name.

Monitor data for the topology

Each server has a monitor that exposes that server's view of the topology in its monitor backend, so that peer servers can periodically read this information to determine if there are changes in the topology. Topology data includes the following:

- The server ID of the current master, if the master is not known.
- The instance name of the current master, or if a master is not set, a description stating why a master is not set.
- A flag indicating if this server thinks that it should be the master.
- A flag indicating if this server is the current master.
- A flag indicating if this server was forced as master.
- The total number of configured peers in the topology group.
- The peers connected to this server.
- The current availability of this server
- A flag indicating whether or not this server is not synchronized with its master, or another node in the topology if the master is unknown.
- The amount of time in milliseconds where multiple masters were detected by this server.
- The amount of time in milliseconds where no suitable server is found to act as master.
- A SHA-256 digest encoded as a base-64 string for the current subtree contents.

The following metrics are included if this server has processed any operations as master:

- The number of operations processed by this server as master.
- The number of operations processed by this server as master that were successful.
- The number of operations processed by this server as master that failed to validate.
- The number of operations processed by this server as master that failed to apply.
- The average amount of time taken (in milliseconds) by this server to process operations as the master.
- The maximum amount of time taken (in milliseconds) by this server to process an operation as the master.

Updating the server instance listener certificate

To change the SSL certificate for the server, update the keystore and truststore files with the new certificate. The certificate file must have the new certificate in PEM-encoded format, such as:

```
-----BEGIN CERTIFICATE-----
```

```
MIIDKTCCAhGgAwIBAgIEacgGrDANBgkqhkiG9w0BAQsFADBFMR4wHAYDVQQKExVVbmJvdW5kSUQgQ2
VydGlmaWNhdGUxIzAhBgNVBAMTGnZtLW1lZGl1bS03My51bmJvdW5kaWQubGFiMB4XDTE1MTAxMjE1
MzU0OFoXDTM1MTAwNzE1MzU0OFowRTEeMBwGA1UEChMVVW5ib3VuZElEIENlcnRpZmljYXRlMSMwIQ
YDVQQDExp2bS1tZWRpdW0tNzMudW5ib3VuZGlkLmxhYjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBAKN4tAN3o9Yw6Cr9hivwVDxJqF6+aEi9Ir3WGFYLSrggRNXsiAOfWkSMWdIC5vyF5OJ9Dl
IgvHL4OuqP/YNEGzKDkgr6MwtUeVSK14+dCixygJGC0nY7k+f0WSCjtIHzrmc4WWdrZXmgb+qv9Lup
S30JG0FXtcbGkYpjaKXIEqMg4ekz3B5cAvE0SQUFyXEdN4rWOn96nVFkb2CstbiPzAgne2tu7paJ6S
GFOW0UF7v018XY1m2WHBIoD0WC8nOVLTG9zFUavaOxtlt1TlhClkI4HRMNg8n2EtSTdQRizKuw9DdT
XJBb6Kfvnp/nI73VHRyt47wUVueehEDfLtDP8pMCAwEAAaMhMB8wHQYDVR0OBBYEFMrwjWxl2K+yd9
+Y65oKn0g5jITgMA0GCSqGSIb3DQEBCwUAA4IBAQBpsBYodblUGew+HewqtO2i8Wt+vAbt31zM5/kR
vo6/+iPEASTvZdCzIBcgletxKGKeCQ0GPeHr42+erakiwmGDlUTYrU3LU5pTGTDLuR2IllTT5xlEhC
WJGWipW4q3Pl3cX/9m2ffY/JLYDfTJaoJvnXrh7Sg719skkHjWZQgOHXlkPLx5TxFGhAovE1D4qLVR
WGohdpWDrIgFh0DVfoyAn1Ws9ICCXdRayajFI4Lc6K1m6SA5+25Y9nno8BhVPf4q5OW6+UDc8MsLbB
sxpwvR6RJ5cv3ypfOriTehJsG+9ZDo7YeqVsTVGwAlW3PiSd9bYP/8yu9Cy+0MfcWcSeAE
-----END CERTIFICATE-----
```

If clients that already have a secure connection established with this server need to be maintained, information about both certificates can reside in the same file (each with their own begin and end headers and footers).

After the keystore and truststore files are updated, run the following `dsconfig` command to update the server's certificate in the topology registry:

```
$ bin/dsconfig set-server-instance-listener-prop \
  --instance-name <server-instance-name> \
  --listener-name ldap-listener-mirrored-config \
  --set listener-certificate <path-to-new-certificate-file>
```

The `listener-certificate` in the topology registry is like a trust store. The public certificates that it has are automatically trusted by the local server. When the local server attempts a secure LDAP connection to a peer, and the peer presents it with its certificate, the local server will check the `listener-certificate` property for that server in the topology registry. If the property contains the peer server's certificate, the local server will trust the peer.

Remove the self-signed certificate

The server is installed with a self-signed certificate and key (`ads-certificate`), which are used for internal purposes such as replication authentication, inter-server authentication in the topology registry, reversible password encryption, and encrypted backup/LDIF export. The ads-certificate lives in the keystore file called `ads-truststore` under the server's `/config` directory. If your deployment requires removing the self-signed certificate, it can be replaced.

The certificate is stored in the topology registry, which enables replacing it on one server and having it mirrored to all other servers in the topology. Any change is automatically mirrored on other servers in the topology. It is stored in human-readable PEM-encoded format and can be updated with `dsconfig`. The following general steps are required to replace the self-signed certificate:

1. Prepare a new keystore with the replacement key-pair.
2. Update the server configuration to use the new certificate by adding it to the server's list of certificates in the topology registry so that it is trusted by other servers.
3. Update the server's `ads-truststore` file to use the new key-pair.
4. Retire the old certificate by removing it from the topology registry.

Note
Replacing the entire key-pair instead of just the certificate associated with the original private key can make existing backups and LDIF exports invalid. This should be performed immediately after setup or before the key-pair is used. After the first time, only the certificate associated with the private key should have to be changed, for example, to extend its validity period or replace it with a certificate signed by a different CA.

Prepare a new keystore with the replacement key-pair

The self-signed certificate can be replaced with an existing key-pair, or the certificate associated with the original key-pair can be used.

Use an existing key-pair

If a private key and certificate(s) in PEM-encoded format already exist, both the original private key and self-signed certificate can be replaced in ads-truststore with the `manage-certificates` tool. The following command imports existing certificates into a new keystore file, `ads-truststore.new`:

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file existing.key \
  --certificate-file existing.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

The certificates listed using the `--certificate-file` options must be ordered so that each subsequent certificate is the issuer for the previous one. So the server certificate comes first, the intermediate certificates next (if any), and the root CA certificate last.

Use the certificate associated with the original key-pair

The certificate associated with the original server-generated private key can be replaced with the following commands:

1. Create a CSR for the `ads-certificate`:

   ```
   $ bin/manage-certificates generate-certificate-signing-request \
     --keystore ads-truststore \
     --keystore-type JKS \
     --keystore-password-file ads-truststore.pin \
     --alias ads-certificate \
   ```

```
  --use-existing-key-pair \
  --subject-dn "CN=ldap.example.com,O=Example Corporation,C=US" \
  --output-file ads.csr
```

2. Submit `ads.csr` to a CA for signing.
3. Export the server’s private key into `ads.key`:

```
$ bin/manage-certificates export-private-key \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --output-file ads.key
```

4. Import the certificates obtained from the CA (the CA-signed server certificate, any intermediate certificates, and root CA certificate) into `ads-truststore.new`:

```
$ bin/manage-certificates import-certificate \
  --keystore ads-truststore.new \
  --keystore-type JKS \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --private-key-file ads.key \
  --certificate-file new-ads.crt \
  --certificate-file intermediate.crt \
  --certificate-file root-ca.crt
```

Update the server configuration to use the new certificate

To update the server to use the desired key-pair, the `inter-server-certificate` property for the server instance must first be updated in the topology registry. The old and the new certificates may appear within their own begin and end headers in the `inter-server-certificate` property to support transitioning from the old certificate to the new one.

1. Export the server’s old ads-certificate into `old-ads.crt`:

```
$ bin/manage-certificates export-certificate \
  --keystore ads-truststore \
  --keystore-password-file ads-truststore.pin \
  --alias ads-certificate \
  --export-certificate-chain \
  --output-file old-ads.crt
```

2. Concatenate the old, new certificate, and issuer certificates into one file. On Windows, an editor like notepad can be used. On Unix platforms, use the following command:

```
$ cat old-ads.crt new-ads.crt intermediate.crt root-ca.crt > chain.crt
```

3. Update the `inter-server-certificate` property for the server instance in the topology registry using `dsconfig`:

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name <instance-name> \
  --set “inter-server-certificate<chain.crt”
```

Update the ads-truststore file to use the new key-pair

The server will still use the old `ads-certificate`. When the new `ads-certificate` needs to go into effect, the old `ads-truststore` file must be replaced with `ads-truststore.new` in the server's `config` directory.

```
$ mv ads-truststore.new ads-truststore
```

Retire the old certificate

The old certificate is retired by removing it from the topology registry when it has expired. All existing encrypted backups and LDIF exports are not affected because the public key in the old and new server certificates are the same, and the private key will be able to decrypt them.

```
$ cat new-ads.crt intermediate.crt root-ca.crt > chain.crt
```

```
$ bin/dsconfig -n set-server-instance-prop \
  --instance-name <instance-name> \
  --set “inter-server-certificate<chain.crt”
```

Domain Name Service (DNS) caching

If needed, two global configuration properties can be used to control the caching of hostname-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system. Use the `dsconfig` tool to configure these properties.

network-address-cache-ttl– Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a hostname-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.

network-address-outage-cache-enabled – Caches hostname-to-IP address results in the event of a DNS outage. This is set to `true` by default, meaning name resolution results are cached. Unexpected service interruptions may occur during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

IP address reverse name lookups

Ping Identity servers do not explicitly perform numeric IP address-to-hostname lookups. However, address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing may trigger implicit reverse name lookups. For more information about how address masks are configured in the server, review the following information for each server:

- ACI dns: bind rules under *Managing Access Control* (Directory Server and PingDirectory Proxy Servers)

- `ds-auth-allowed-address`: *Adding Operational Attributes that Restrict Authentication* (Directory Server)
- Connection Criteria: *Restricting Server Access Based on Client IP Address* (Directory Server and PingDirectory Proxy Servers)
- Connection Handlers: restrict server access using Connection Handlers (Configuration Reference Guide for all PingData servers)

Configure the synchronization environment with dsconfig

The `dsconfig` tool can be used to configure any part of the PingDataSync Server, but will likely be used for more fine-grained adjustments. If configuring a Sync Pipe for the first time, use the `bin/create-sync-pipe-config` tool to guide through the necessary Sync Pipes creation steps.

Configure server groups with dsconfig interactive

In a typical deployment, onePingDataSync Server and one or more redundant failover servers are configured. Primary and secondary servers must have the same configuration settings to ensure proper operation. To enable this, assign all servers to a server group using the `dsconfig` tool. Any change to one server will automatically be applied to the other servers in the group.

Run the `dsconfig` command and set the global configuration property for server groups to `all-servers`. On the primary PingDataSync Server, do the following:

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:all-servers
```

Updates to servers in the group are made using the `--applyChangeTo servergroup` option of the `dsconfig` command. To apply the change to one server in the group, use the `--applyChangeTo single-server` option. If additional servers are added to the topology, the `setup` tool will copy the configuration from the primary server to the new server(s).

Start the Global Sync Configuration with dsconfig interactive

After the Synchronization topology is configured, perform the following steps to start the Global Sync Configuration, which will use only those Sync Pipes that have been started:

1. On the `dsconfig` main menu, type the number corresponding to the Global Sync Configuration.

2. On the Global Sync Configuration menu, type the number corresponding to view and edit the configuration.

3. On the Global Sync Configuration Properties menu, type the number corresponding to setting the started property, and then follow the prompts to set the value to `TRUE`.

4. On the Global Sync Configuration Properties menu, type **f** to save and apply the changes.

Prepare external server communication

The `prepare-endpoint-server` tool sets up any communication variances that may occur between the PingDataSync Server and the external servers. Typically, directory servers can have different security settings, privileges, and passwords configured on the Sync Source that might reject the import of entries in the Sync Destination.

The `prepare-endpoint-server` tool also creates a Sync User Account and its privileges on all of the external servers (see About the Sync User Account for more detailed information). The `prepare-endpoint-server` tool verifies that the account has the proper privileges to access the `firstChangeNumber` and `lastChangeNumber` attributes in the root DSE entry so that it can access the latest changes. If the Sync User does not have the proper privileges, the PingDataSync Server displays a warning message, which is saved in the `logs/prepare-endpoint-server.log` file.

> **Note**
> If the synchronization topology was created using the `create-sync-pipe-config` tool, this command does not need to be run. It is already part of the `create-sync-pipe-config` process.

Perform the following steps to prepare thePingDataSync Server for external server communication:

1. Use the `prepare-endpoint-server` tool to prepare the directory server instances on the remote host for synchronization as a data source for the subtree, `dc=example,dc=com`. If the user account is not present on the external server, it will be created if a parent entry exists.

```
$ bin/prepare-endpoint-server \
  --hostname sun-ds1.example.com \
  --port 21389 \
  --syncServerBindDN "cn=Sync User,dc=example,dc=com" \
  --syncServerBindPassword secret \
  --baseDN "dc=example,dc=com" \
  --isSource
```

2. When prompted, enter the bind DN and password to create the user account. This step enables the change log database and sets the `changelog-maximum-age` property.
3. Repeat steps 1–2 for any other external source servers.
4. For the destination servers, repeat steps 2–3 and include the `--isDestination` option. If destination servers do not have any entries, a "Denied" message will display when creating the `cn=Sync User` entry.

```
$ bin/prepare-endpoint-server \
  --hostname PingIdentity-ds1.example.com \
  --port 33389 \
  --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
  --syncServerBindPassword sync \
--baseDN "dc=example,dc=com" \
--isDestination
```

5. Repeat step 4 for any other destination servers.

Configuration with the dsconfig tool

The Ping Identity servers provide several command-line tools for management and administration. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

The `dsconfig` tool is the text-based management tool used to configure the underlying server configuration. The tool has three operational modes:

- Interactive mode
- Non-interactive mode
- Batch mode

The `dsconfig` tool also offers an offline mode using the `--offline` option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. View detailed argument options and examples by typing `--help` with the command.

```
$ bin/dsconfig --help
```

To list the subcommands for each command:

```
$ bin/dsconfig --help-subcommands
```

To list more detailed subcommand information:

```
$ bin/dsconfig list-log-publishers --help
```

Use dsconfig in interactive mode

Running `dsconfig` in interactive command-line mode provides a menu-driven interface for accessing and configuring the PingData server. To start `dsconfig` in interactive mode, run the tool without any arguments:

```
$ bin/dsconfig
```

Running the tool requires server connection and authentication information. After connection information is confirmed, a menu of the available operation types is displayed.

Use dsconfig in non-interactive mode

Non-interactive command-line mode provides a simple way to make arbitrary changes to the server, and to use administrative scripts to automate configuration changes. To make changes to multiple configuration objects at the same time, use batch mode.

The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The `--no-prompt` argument specifies non-interactive mode. The `{globalArgs}` argument provides a set of arguments that specify how to connect and authenticate to the server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on the implementation. The `{subcommand}` specifies which general action to perform. The following uses standard LDAP connections:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

The following uses SASL GSSAPI (Kerberos) parameters:

```
$ bin/dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The `{subcommandArgs}` argument contains a set of arguments specific to the particular task. To always display the advanced properties, use the `--advanced` command-line option.

> **Note**
> Global arguments can appear anywhere on the command line. The subcommand-specific arguments can appear anywhere after the subcommand.

Use dsconfig batch mode

The `dsconfig` tool provides a batching mechanism that reads multiple invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The `--no-prompt` option is required with `dsconfig` in batch mode.

```
$ bin/dsconfig --no-prompt \
  --hostname host1 \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --batch-file /path/to/sync-pipe-config.txt
```

If a dsconfig command has a missing or incorrect argument, the command will fail and stop the batch process without applying any changes to the server. A --batch-continue-on-error option is available, which instructs dsconfig to apply all changes and skip any errors.

View the logs/config-audit.log file to review the configuration changes made to the server, and use them in the batch file. The batch file can have blank lines for spacing, and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The PingDataSync Server also provides a docs/sun-ds-compatibility.dsconfig file for migrations from Sun/Oracle to Ping Identity PingDataSync Server machines.

Using the resync Tool

The resync tool provides bulk synchronization that can be used to verify the synchronization setup. The tool operates on a single Sync Pipe at a time, retrieves entries from the Sync Source in bulk, and compares the source entries with the corresponding destination entries. If destination entries are missing or attributes are changed, they are updated.

The command provides a --dry-run option that can be used to test the matches between the Sync Source and Destination, without committing any changes to the target topology. The resync tool also provides options to write debugging output to a log.

> **Note**
> The resync tool should be used for relatively small datasets. For large deployments, export entries from the Sync Source into an LDIF file, run the bin/translate-ldif tool to translate and filter the entries into the destination format, and then import the result LDIF file into the Sync Destination.

Use the resync --help command for more information and examples. Logging is located in logs/tools/resync.log and logs/tools/resync-errors.log. If necessary, the logging location can be changed with the --logFilePath option.

Testing Attribute and DN Maps

The resync tool can be used to test how attribute maps and DN maps are configured by synchronizing a single entry. If the --logFilePath and --logLevel options are specified, the resync tool generates a log file with details.

Use the--dry-run option and specify a single entry, such as uid=user.0. Any logging performed during the operation appears in logs/tools/resync.log.

```
$ bin/resync --pipe-name sun-to-ds-sync-pipe \
  --sourceSearchFilter "(uid=user.0)" \
  --dry-run \
  --logLevel debug
```

Verifying the Synchronization Configuration

The most common use for the resync tool is to test that the Sync Pipe configuration has been set up correctly. For example, the following procedure assumes that the configuration was set up with the Sync Source topology (two replicated Sun Directory servers) with 2003 entries; the

Sync Destination topology (two replicated PingData Directory Server) has only the base entry and the `cn=Sync User` entry. Both source and destination topologies have their LDAP Change Logs enabled. Because both topologies are not actively being updated, the `resync` tool can be run with one pass through the entries.

Use `resync` with the `--dry-run` option to check the synchronization configuration. The output displays a timestamp that can be tracked in the logs.

```
$ bin/resync --pipe-name sun-to-ds-sync-pipe \
  --numPasses 1 \
  --dry-run
```

```
Starting Pass 1

Status after completing all passes[20/Mar/2010:10:20:07 -0500]
----------------------------------
Source entries retrieved 2003
Entries missing 2002
Entries out-of-sync 1
Duration (seconds) 4

Resync completed in 4 s.

0 entries were in-sync, 0 entries were modified, 0 entries were created,
1 entries are still out-of-sync, 2002 entries are still missing, and
0 entries could not be processed due to an error
```

Populating an Empty Sync Destination Topology

The following procedure uses the `resync` tool to populate an empty Sync Destination topology for small datasets. For large deployments, use the `bin/translate-ldif`.

In this example, assume that the Sync Destination topology has only the base entry (`dc=example,dc=com`) and the `cn=Sync User` entry. Perform the following steps to populate an empty Sync Destination:

1. Run the `resync` command with the log file path and with the log level `debug`. Logging is located in `logs/tools/resync.log` and `logs/tools/resync-errors.log`.

   ```
   $ bin/resync --pipe-name sun-to-ds-sync-pipe \
     --numPasses 1 \
     --logLevel debug
   ```

2. Open the `logs/resync-failed-DNs.log` file in a text editor to locate the error and fix it. An entry cannot be created because the parent entry does not exist.

   ```
   # Entry '(see below)' was dropped because there was a failure at the
   resource:
   Failed to create entry uid=mlott,ou=People,dc=example,dc=com. Cause:
   LDAPException(resultCode=no such object, errorMessage='Entry
   uid=user.38,ou=People,dc=example,dc=com cannot be added because its parent
   entry ou=People,dc=example,dc=com does not exist in the server',
   ```

```
matchedDN='dc=example,dc=com')
(id=1893859385ResourceOperationFailedException.java:126 Build
revision=4881)
dn: uid=user.38,ou=People,dc=example,dc=com
```

3. Rerun the `resync` command. The command creates the entries in the Sync Destination topology that are present in the Sync Source topology.

```
$ bin/resync --pipe-name sun-to-ds-sync-pipe
```

```
...(output from each pass)...

Status after completing all passes[20/Mar/2016:14:23:33 -0500]
-----------------------------
Source entries retrieved 160
Entries in-sync 156
Entries created 4
Duration (seconds) 11

Resync completed in 12 s.

156 entries were in-sync, 0 entries were modified, 4 entries were created,
0 entries are still out-of-sync, 0 entries are still missing, and 0
entries could not be processed due to an error
```

Note
If importing a large amount of data into an PingData Directory Server, run `export-ldif` and `import-ldif` on the newly populated backend for most efficient disk space use. If needed, run `dsreplication initialize` to propagate the efficient layout to additional replicas.

Setting the Synchronization Rate

The `resync` command has a `--ratePerSecondFile` option that enables a specific synchronization rate. The option can be used to adjust the rate during off-peak hours, or adjust the rate based on measured loads for very long operations.

Note
The `resync` command also has a `--ratePerSecond` option. If this option is not provided, the tool operates at the maximum rate.

Run the `resync` tool first at 100 operations per second, measure the impact on the source servers, then adjust as desired. The file must contain a single positive integer number surrounded by white space. If the file is updated with an invalid number, the rate is not updated.

1. Create a text file containing the rate. The number must be a positive integer surrounded by white space.

```
$ echo ’100 ’ > rate.txt
```

2. Run the `resync` command with the `--ratePerSecondFile` option.

```
$ bin/resync --pipe-name "sun-to-ds-sync-pipe" \
  --ratePerSecondPath rate.txt
```

Synchronizing a Specific List of DNs

The `resync` command enables synchronizing a specific set of DNs that are read from a file using the `--sourceInputFile` option. This option is useful for large datasets that require faster processing by targeting individual base-level searches for each source DN in the file. If any DN fails (parsing, search, or process errors), the command creates an output file of the skipped entries (`resync-failed-DNs.log`), which can be run again.

The file must contain only a list of DNs in LDIF format with `dn:` or `dn::`. The file can include comment lines. All DNs can be wrapped and are assumed to be wrapped on any lines that begin with a space followed by text. Empty lines are ignored.

Small files can be created manually. For large files, use `ldapsearch` to create an LDIF file, as follows:

1. Run an `ldapsearch` command using the special OID "1.1" extension, which only returns the DNs in the DIT. For example, on the Sync Source directory server, run the following command:

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com \
  --baseDN dc=example,dc=com \
  --searchScope sub "(objectclass=*)" "1.1" > dn.ldif
```

2. Run the `resync` command with the file.

```
$ bin/resync --pipe-name "sun-to-ds-pipe" \
  --sourceInputFile dn.ldif
```

```
Starting pass 1
[20/Mar/2016:10:32:11 -0500]

-------------------------------------
Resync pass 1
Source entries retrieved 1999
Entries created 981
Current pass, entries processed 981
Duration (seconds) 10
Average ops/second 98
Status after completing all passes[20/Mar/2016:10:32:18 -0500]

----------------------------------
Source entries retrieved 2003
Entries created 2003
Duration (seconds) 16
Average ops/second 98
Resync completed in 16 s.
0 entries were in-sync, 0 entries were modified, 2003 entries were
created, 0 entries are still out-of-sync, 0 entries are still missing, and
0 entries could not be processed due to an error
```

3. View the `logs/tools/resync-failed-DNs.log` to determine skipped DNs. Correct the source DNs file, and rerun the `resync` command.

Using the realtime-sync Tool

The `bin/realtime-sync` tool controls starting and stopping synchronization globally or for individual Sync Pipes. The tool also provides features to set a specific starting point for real-time synchronization.

To see the current status of real-time synchronization, view the monitor properties: `num-sync-ops-in-flight`, `num-ops-in-queue`, and `source-unretrieved-changes`. For example, use `ldapsearch` to view a specific Sync Pipe’s monitor information:

```
$ bin/ldapsearch --baseDN cn=monitor \
--searchScope sub "(cn=Sync PipeMonitor: PIPE_NAME)"
```

The Stats Logger can also be used to view status. See the *Ping IdentityDirectory Server Administration Guide* for details.

Starting Real Time Synchronization Globally

The `realtime-sync` tool assumes that the synchronization topology is configured correctly.

Perform the following steps to start real time synchronization globally:

1. Run the tool from the `<server-root>/bin` directory. This example assumes that a single Sync Pipe called "dsee-to-ds-sync-pipe" exists.

```
$ bin/realtime-sync start --pipe-name "dsee-to-ds-sync-pipe" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. If more than one Sync Pipe is configured, specify each using the `--pipe-name` option. The following example starts synchronization for a bidirectional synchronization topology.

```
$ bin/realtime-sync start --pipe-name "Sun DS to DS" \
  --pipe-name "DS to Sun DS" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Starting or Pausing Synchronization

Pause or start synchronization by using the `start` and `stop` subcommands. If synchronization is stopped and then restarted, changes made at the Sync Source while synchronization was stopped will still be detected and applied. Synchronization for individual Sync Pipes can be started or stopped using the `--pipe-name` argument. If the `--pipe-name` argument is omitted, then synchronization is started or stopped globally.

The following command stops all Sync Pipes:

```
$ bin/realtime-sync stop --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --no-prompt
```

If a topology has two Sync Pipes, Sync Pipe1 and Sync Pipe2, the following command stops Sync Pipe1.

```
$ bin/realtime-sync stop --pipe-name "Sync Pipe1" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

Setting Startpoints

Startpoints instruct the Sync Pipe to ignore all changes made prior to the current time. Once synchronization is started, only changes made after this command is run will be detected at the Sync Source and applied at the Sync Destination.

The `set-startpoint` subcommand is often run during the initial setup prior to starting realtime synchronization. It should be run prior to initializing the data in the Sync Destination.

The `set-startpoint` subcommand can start synchronization at a specific change log number, or back at a state that occurred at a specific time. For example, synchronization can start 10 minutes prior to the current time.

Perform the following steps to set a synchronization startpoint:

1. If started, stop the synchronization topology globally with the following command:

   ```
   $ bin/realtime-sync stop --pipe-name "Sync Pipe1" \
     --port 389 \
     --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret \
     --no-prompt
   ```

2. Set the startpoint for the synchronization topology. Any changes made before setting this command will be ignored.

   ```
   $ bin/realtime-sync set-startpoint --pipe-name "Sync Pipe1" \
     --port 389 \
     --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret \
     --no-prompt \
     --beginning-of-changelog
   ```

   ```
   Set StartPoint task 2011072109564107 scheduled to start immediately
   [21/Jul/2016:09:56:41 -0500] severity="INFORMATION" msgCount=0
   msgID=1889535170
   message="The startpoint has been set for Sync Pipe 'Sync Pipe1'.
   Synchronization will resume from the last change number in the Sync
   ```

```
Source"
Set StartPoint task 2011072109564107 has been successfully completed
```

Restarting Synchronization at a Specific Change Log Event

Perform the following steps to restart synchronization at a specific event:

1. Search for a specific change log event from which to restart the synchronization state. On one of the endpoint servers, run `ldapsearch` to search the change log.

```
$ bin/ldapsearch -p 1389
--bindDN "uid=admin,dc=example,dc=com" \
--bindPassword secret \
--baseDN cn=changelog \
--dontWrap
```

```
"(objectclass=*)"
dn: cn=changelog
objectClass: top
objectClass: untypedObject
cn: changelog

dn: changeNumber=1,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.13,ou=People,dc=example,dc=com
changeType: modify
changes::
cmVwbGFjZTogcm9vbU51bWJlcgpyb29tTnVtYmVyOiAwMTM4Ci0KcmVwbGFjZTogbW9kaW
ZpZXJzTmFtZQptb2RpZmllcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290
IEROcyxjbj1jb25maWcKLQpyZXBsYWNlOiBkcy11cGRhdGUtdGltZQpkcy11cGRhdGUtdG
ltZTo6IEFBQUJKZ25OWlUwPQotCgA=
changenumber: 1
          ... (more output)
dn: changeNumber=2329,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.49,ou=People,dc=example,dc=com
changeType: modify
changes::
cmVwbGFjZTogcm9vbU51bWJlcgpyb29tTnVtYmVyOiAwNDMzCi0KcmVwbGFjZTogbW9kaW
ZpZXJzTmFtZQptb2RpZmllcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290
IEROcyxjbj1jb25maWcKLQpyZXBsYWNlOiBkcy11cGRhdGUtdGltZQpkcy11cGRhdGUtdG
ltZTo6IEFBQUJKZ25OMC84PQotCgA=
changenumber: 2329
```

2. Restart synchronization from change number 2329 using the `realtime-sync` tool. Any event before this change number will not be synchronized to the target endpoint.

```
$ bin/realtime-sync set-startpoint \
  --change-number 2329 \
  --pipe-name "Sync Pipe 1" \
  --bindPassword secret \
  --no-prompt
```

Changing the Synchronization State by a Specific Time Duration

The following command will start synchronizing data at the state that occurred 2 hours and 30 minutes prior to the current time on External Server 1 for Sync Pipe 1. Any changes made before this time will not be synchronized. Specify days (d), hours (h), minutes (m), seconds (s), or milliseconds (ms).

Use `realtime-sync` with the `--startpoint-rewind` option to set the synchronization state and begin synchronizing at the specified time.

```
$ bin/realtime-sync set-startpoint \
  --startpoint-rewind 2h30m \
  --pipe-name "Sync Pipe 1" \
  --bindPassword secret \
  --no-prompt
```

Scheduling a Realtime Sync as a Task

The `realtime-sync` tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDataSync Server's process. To schedule an operation, supply LDAP connection options that allow this tool to communicate with the server through its task interface. Tasks can be scheduled to run immediately or at a later time. Once scheduled, tasks can be managed using the `manage-tasks` tool.

Perform the following steps to schedule a synchronization task:

1. Use the `--start` option with the `realtime-sync` command to schedule a start for the synchronization topology. The following command will set the start time at July 21, 2016 at 12:01:00 AM. The scheduled task can be stopped with the `--stop` subcommand.

```
$ bin/realtime-sync set-startpoint \
  --pipe-name "sun-to-ds-sync-pipe" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --start 20150721000100 \
  --no-prompt
```

```
Set StartPoint task 2009072016103807 scheduled to start Jul 21, 2016
12:01:00 AM CDT
```

2. Run the `manage-tasks` tool to manage or cancel the task.

```
$ bin/manage-tasks --port 7389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configuring the Directory Server Backend for Synchronizing Deletes

The Directory Server's change log backend's `changelog-deleted-entry-include-attribute` property specifies which attributes should be recorded in the change log entry during a DELETE operation. Normally, the PingDataSync Server cannot correlate a deleted entry to the entry on the destination. If a Sync Class is configured with a filter, such as "`include-filter: objectClass=person`," the `objectClass` attribute must be recorded in the change log entry. Special correlation attributes (other than DN), will also need to be recorded on the change log entry to be properly synchronized at the endpoint server.

On each Directory Server backend, use the `dsconfig` command to set the property.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set changelog-deleted-entry-include-attribute:objectClass
```

If the destination endpoint is an Oracle/Sun DSEE (or Sun DS) server, the Sun DSEE server does not store the value of the user deleting the entry, specified in the modifiers name attribute. It only stores the value of the user who last modified the entry while it still existed.

To set up a Sun DSEE destination endpoint to record the user who deleted the entry, use the Ping Identity Server SDK to create a plug-in as follows:

1. Update the Sun DSEE schema to include a `deleted-by-sync auxiliary` objectclass. It will only be used as a marker objectclass, and not require or allow additional attributes to be present on an entry.
2. Update the Sun DSEE Retro Change Log Plug-in to include the `deleted-by-sync auxiliary` object class as a value for the `deletedEntryAttrs` attribute.
3. Write an `LDAPSyncDestinationPlugin` script that in the `preDelete()` method modifies the entry that is being deleted to include the `deleted-by-sync` objectclass.
4. Update the Sync Class filter that is excluding changes by the Sync User to also include `(!(objectclass=deleted-by-sync))`.

Configure DN maps

Similar to attribute maps, DN maps define mappings when destination DNs differ from source DNs. These differences must be resolved using DN maps in order for synchronization to successfully take place. For example, the Sync Source could have a DN in the following format:

```
uid=jdoe,ou=People,dc=example,dc=com
```

While the Sync Destination could have the standard X.500 DN format.

DN mappings allow the use of wild cards for DN transformations. A single wild card (`*`) matches a single RDN component and can be used any number of times. The double wild card (`**`) matches zero or more RDN components and can be used only once.

Note
If a literal '*' is required in a DN then it must be escaped as '\2A'.

The wild card values can be used in the `to-dn-pattern` attribute using `{1}` to replace their original index position in the pattern, or `{attr}` to match an attribute value. For example:

```
*,**,dc=com->{1},ou=012,o=example,c=us
```

For example, using the DN, `uid=johndoe,ou=People,dc=example,dc=com`, and mapping to the target DN, `uid=johndoe,ou=012,o=example,c=us`:

- "*" matches one RDN component, `uid=johndoe`
- "**" matches zero or more RDN components, `ou=People,dc=example`
- "dc=com" matches `dc=com` in the DN.

The DN is mapped to the `{1},ou=012,o=example,c=us`. "{1}" substitutes the first wildcard element "uid=johndoe", so that the DN is successfully mapped to:

```
uid=johndoe,ou=012,o=example,c=us
```

Regular expressions and attributes from the user entry can also be used in the `to-dn-pattern` attribute. For example, the following expression constructs a value for the `uid` attribute, which is the RDN, out of the initials (first letter of givenname and `sn`) and the employee ID (the `eid` attribute) of a user.

```
uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid},{2},o=example
```

Note
The PingDataSync Server automatically validates any DN mapping prior to applying the configuration.

Configuring a DN Map Using dsconfig

A DN map can be configured using `dsconfig`, either with the interactive DN Map menu, or from the command line.

Perform the following to configure a DN map:

1. Use `dsconfig` to create a DN map for the PingDataSync Server.

```
$ bin/dsconfig --no-prompt create-dn-map \
  --map-name nested-to-flattened \
  --set "from-dn-pattern:*,*,dc=example,dc=com" \
  --set "to-dn-pattern:uid={givenname:/^(.)(.*)/\$1/s}{sn:/^(.)(.*)/\$1/s}
(eid},{2},o=example" \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. After DN mappings are configured, add the new DN map to a new Sync Class or modify an existing Sync Class.

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name test-sync-pipe \
  --class-name test-sync-class \
  --set dn-map:test-dn-map \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configure synchronization with JSON attribute values

The PingDataSync Server supports synchronization of attributes that hold JSON objects. The following scenarios are supported:

- **Synchronizing a JSON attribute to another JSON attribute** - A subset of fields can be synchronized, optionally retaining fields that appear at the destination but not at the source.
- **Synchronizing a JSON attribute to a non-JSON attribute** - A single field of the JSON value can be extracted with a constructed attribute mapping.
- **Synchronizing a non-JSON attribute to a JSON attribute** - The source value can be escaped so that it ensures the JSON value is properly formatted.
- **Attribute correlation** - A JSON field can be used when correlating a destination entry with a source entry.

The following examples show configuration scenarios based on the LDAP `ubidEmailJSON` attribute, which has fields of `value`, `type`, `primary`, and `verified`:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "primary" : true,
                "verified" : true}
```

Synchronize ubidEmailJSON fully

If a source JSON attribute value should be synchronized fully to a destination JSON attribute value, no special configuration is required.

Synchronize a subset of fields from the source attribute

For example, the following configuration can be used to synchronize the `value` and `type` fields of `ubidEmailJSON` from the source to a destination. To synchronize this source value:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "primary" : true}
```

to this value at the destination:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home"}
```

A JSON Attribute configuration object must be created and associated with the Sync Class. This can be done by either explicitly including the fields to synchronize:

```
$ bin/dsconfig create-json-attribute --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set include-field:type \
  --set include-field:value
```

Or by excluding the fields that should not be synchronized:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set exclude-field:preferred \
  --set exclude-field:verified
```

If the destination is prepared to only handle a specific subset of fields, then list the fields to include. However, if only a small, known subset of fields from the source should be excluded, then `exclude-field` could be used. In this example, the destination data for the `ubidEmailJSON` attribute will always be a subset of the full data.

> **Note**
> A Sync Class can be configured to exclude certain attributes from synchronization. Creating a regular attribute mapping will override this setting, and the attribute will be synchronized. Creating a JSON attribute mapping does not override this setting, and the JSON attribute will not be synchronized. A JSON attribute is not a traditional attribute mapping. It only includes information on the destination attribute name. To work around this, the attribute either needs to be mapped from a source attribute, or have its value constructed.

The following scenario illustrates how the destination can include additional fields that are not present at the source.

Retain destination-only fields

To synchronize changes to the source fields while preserving the value of the the `verified` field of the `ubidEmailJSON` attribute at the destination, configure the JSON Attribute as follows:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set id-field:value \
  --set exclude-field:verified
```

The `verified` field is excluded and `value` is chosen to correlate destination values with source values. For example, given that the source and destination `value` fields match, if the source initially contained:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home"}
```

and the destination contained:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "verified" : true},
```

if the source changed to:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "other"}
```

then the destination would change to:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "other",
                "verified" : true}
```

However, if the source changed to:

```
ubidEmailJSON: {"value" : "john.smith@example.com",
                "type" : "home"}
```

then the destination would be updated to:

```
ubidEmailJSON: {"value" : "john.smith@example.com",
                "type" : "home"}
```

The `verified` field has been dropped because this logically represents a new JSON object rather than an update of an existing one.

Synchronize a field of a JSON attribute into a non-JSON attribute

If the source stores:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home"}
```

but the destination stores:

```
mail: jsmith@example.com
```

To synchronize changes between these systems, a constructed attribute mapping must be configured:

```
$ bin/dsconfig create-attribute-mapping \
  --map-name "Attribute Map" \
  --mapping-name mail \
  --type constructed \
  --set "value-pattern:{ubidEmailJSON.value}"
```

The `value-pattern` syntax allows attributes to be referenced by placing them in `{}`. JSON fields within the attribute can be referenced by using the syntax `{attribute.field}`. See this property in the Configuration Refernce guide, or `dsconfig` tool command help for more information.

After the “Attribute Map” is created, it can be referenced from the Sync Class:

```
$ bin/dsconfig set-sync-class-prop
  --pipe-name "A to B" \
  --class-name Users \
  --set "attribute-map:Attribute Map"
```

Note

While LDAP attribute names are not case sensitive, the JSON field names are. By default, errors related to attribute mapping are not logged. To enable error logging, configure the Debug Logger with the following:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true
```

```
$ bin/dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name com.unboundid.directory.sync.mapping \
  --set debug-level:warning
```

Synchronize a non-JSON attribute into a field of a JSON attribute

This is the reverse of the previous example. Suppose the source stores:

```
mail: jsmith@example.com
```

but the destination stores:

```
ubidEmailJSON: {"value" : "jsmith@example.com"}
```

A constructed attribute mapping can be used in this case as well:

```
$ bin/dsconfig create-attribute-mapping \
  --map-name "Attr Map" \
  --mapping-name ubidEmailJSON \
  --type constructed \
  --set 'value-pattern:{{"value" : "{mail:jsonEscape}"}}'
```

When constructing the value, the following are important:

- Double curly brackets (`{{ }}`) are necessary to represent a single curly bracket (`{ }`) in the output. These brackets are typically used to reference attribute values.
- Attribute values that appear within a JSON attribute should be escaped using the `:jsonEscape` modifier. This prevents values that include quotes like `‘“John Smith” <jsmith@example.com>’` from producing invalid JSON.

In this example, a JSON Attribute object should be created since the destination value is likely to be augmented with additional information:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set id-field:value \
  --set include-field:value
```

Correlating attributes based on JSON fields

When the destination of a Sync Pipe is a Ping Directory Server or PingDirectory Proxy Server, source and destination entries can be correlated by referencing a field within a JSON attribute. In the following example, source entries will be matched with destination entries that have the same `value` field within the `ubidEmailJSON` value.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name "A to B" \
  --class-name Users \
  --set destination-correlation-attributes:ubidEmailJSON.value
```

This could also be used with the previous example, which does not store `ubidEmailJSON.value` at the source but maps into it before correlating at the destination.

Configure fractional replication

The PingDataSync Server supports fractional replication to any server type. For example, if a replica only performs user authentications, the PingDataSync Server can be configured to propagate only the `uid` and `userpassword` password policy attributes, reducing the database size at the replica and the network traffic needed to keep this servers synchronized.

The following example configures a fractional replication, where the `uid` and `userPassword` attributes of all entries in the source topology are synchronized to the destination topology. Because the `uid` and `userPassword` attributes are present, the `objectclass` attribute must also be synchronized. The example assumes that a PingDataSync Server and external servers are configured and a Sync Pipe and Sync Class are defined, but realtime synchronization or bulk resync have not been performed.

Perform the following steps to configure fractional replication from the `dsconfig` interactive menu:

1. On the main menu, type the number corresponding to Sync Classes.
2. On the Sync Class menu, type the number corresponding to viewing and editing an existing Sync Class. Assume that only one Sync Class has been defined.
3. Verify that the Sync Pipe and Sync Class exist.
4. On the Sync Class Properties menu, type the number specifying the source LDAP filter (`include-filter` property) that defines which source entries are to be included in the Sync Class.
5. On the Include-Filter Property menu, type the number corresponding to adding a filter value. For this example, type (`objectclass=person`). When prompted, enter another filter. Press **Enter** to continue. On the menu, enter 1 to use the value when specifying it.
6. On the Sync Class Properties menu, type the number corresponding to the `auto-mapped-source-attribute` property. Change the value from "`-all-`" to a specific attribute, so

that only the specified attribute is automatically mapped from the source topology to the destination topology.

7. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to adding the source attributes that will be automatically mapped to the destination attributes of the same name. When prompted, enter each attribute, and then press **Enter**.

```
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: uid
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: userPassword
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: objectclass
Enter another value for the 'auto-mapped-source-attribute' property
[continue]:
```

8. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to removing one or more values. In this example, remove the "`-all-`" value, so that only the `objectclass`, `uid`, and `userPassword` attributes are only synchronized.
9. On the Auto-Mapped-Source-Attribute Property menu, press **Enter** to accept the values.
10. On the Sync Class Properties menu, type the number corresponding to excluding some attributes from the synchronization process. When using the `objectclass=person` filter, the `cn`, `givenName`, and `sn` attributes must be excluded. Enter the option to add one or more attributes, and then add each attribute to exclude on the `excluded-auto-mapped-source-attributes` Property menu. For this example, exclude the `cn`, and `sn` attributes, which are required attributes of the `Person` objectclass. Also exclude the `givenName` attribute, which is an optional attribute of the `inetOrgPerson` objectclass.

```
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]: givenName
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]: sn
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]:
```

11. On the Excluded-Auto-Mapped-Source-Attributes Property menu, press **Enter** to accept the changes.

Note

If using `entryUUID` as a correlation attribute, some attribute uniqueness errors may occur while using the `resync` tool. Either set the `excluded-auto-mapped-source-attributes` property value to `entryUUID` on the Sync Class configuration menu, or run `resync` with the `--excludeDestinationAttr` entryUUID argument.

12. On the Sync Class Properties menu, review the configuration and accept the changes.
13. On the server instances in the destination topology, turn off schema checking to avoid a schema error that occurs when the required attributes in the `Person` object class are not present. Make sure that the global configuration property for the `server-group` is set to `all-servers`. Use the following command to turn off schema checking on all of the servers in the group.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set check-schema:false \
  --applyChangeTo server-group \
  --port 3389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

14. Run `bin/resync` to load the filtered data from the source endpoint to the target endpoint.

```
$ bin/resync --pipe-name "test-sync-pipe" \
  --numPasses 3
```

15. Run `bin/realtime-sync` to start synchronization.

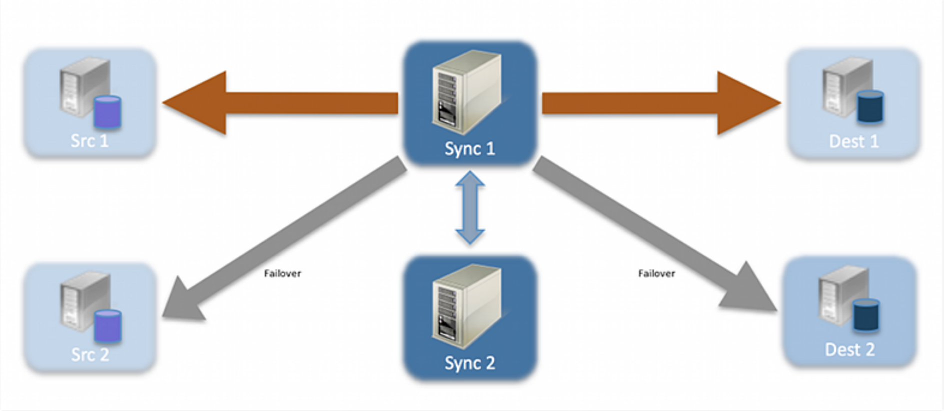
```
$ bin/realtime-sync start --pipe-name "test-sync-pipe" \
  --port 7389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --no-prompt
```

Configure failover behavior

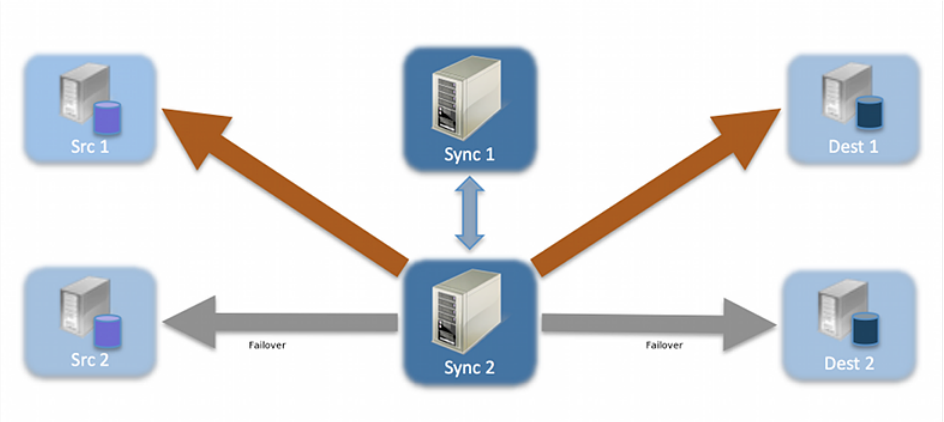
The following illustrates a simplified synchronization topology with a single failover server on the source, destination, and PingDataSync Server, respectively. The gray lines represent possible failover connections in the event the server is down. The external servers are prioritized so that src1 has higher priority than src2; dest1 has higher priority than dest2.

The main PingDataSync Server and its redundant failover instance communicate with each other over LDAP and bind using `cn=IntraSync User,cn=Root DNs,cn=config`. The servers run periodic health checks on each other and share information on all changes that have been processed. Whenever the failover server loses connection to the main server, it assumes that the main server is down and begins processing changes from the last known change. Control reverts back to the main server once it is back online.

Unlike the PingDataSync Server servers, the external servers and their corresponding failover server(s) do not run periodic health checks. If an external server goes offline, the failover server will receive transactions and remain connected to the PingDataSync Server until the Sync Pipe is restarted, regardless of if the main external server comes back online.



The PingDataSync Server in a Simplified Setup



The PingDataSync Server Sample Failover

Conditions that trigger immediate failover

Immediate failover occurs when the PingDataSync Server receives one of the following error codes from an external server:

- BUSY (51)
- UNAVAILABLE (52)
- SERVER CONNECTION CLOSED (81)
- CONNECT ERROR (91)

If the PingDataSync Server attempts a write operation to a target server that returns one of these error codes, the PingDataSync Server will automatically fail over to the next highest prioritized server instance in the target topology, issue an alert, and then reissue the retry attempt. If the operation is unsuccessful for any reason, the server logs an error.

Failover server preference

The PingDataSync Server supports endpoint failover, which is configurable using the `location` property on the external servers. By default, the PingDataSync Server prefers to connect to and failover to endpoint servers in the same location as itself. If no location settings are configured, the PingDataSync Server will iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

The PingDataSync Server does not do periodic health checks and will not failover to a preferred server automatically. Because of the cost of sync failover, it will always stay connected to a given server until that server stops responding or until the Sync Pipe is restarted. When a failover does happen, it will always go back to the most preferred server (optionally using location settings to determine this) and work its way down the list. The following provides an example configuration of external servers:

```
austin1.server.com:1389
london1.server.com:2389
boston1.server.com:3389
austin2.server.com:4389
boston2.server.com:5389
london2.server.com:6389
```

If the austin1 server were to become unavailable, the PingDataSync Server will automatically pick up changes on the next server on the list, london1. If london1 is also down, then the next server, boston1 will be picked up. Once the PingDataSync Server iterates through the list, it returns to the top of the list. So, if the PingDataSync Server is connected to london2 and it goes down, it will fail over to austin1.

To minimize WAN traffic, configure the `location` property for each external server using the `dsconfig` command on the PingDataSync Server. Assume that PingDataSync Server has its own `location` property (set in the Global Configuration) set to "austin."

```
austin1.server.com:1389 location=austin
london1.server.com:2389 location=london
boston1.server.com:3389 location=boston
austin2.server.com:4389 location=austin
boston2.server.com:5389 location=boston
london2.server.com:6389 location=london
```

With the `location` property set for each server, the PingDataSync Server gets its changes from server austin1. If austin1 goes down, the PingDataSync Server will pick up changes from austin2. If austin2 goes down, the server will iterate through the rest of the list in the order it is configured.

The `location` property has another sub-property, `preferred-failover-location` that specifies a set of alternate locations if no servers in this location are available. If multiple values are provided, servers are tried in the order in which the locations are listed. The `preferred-failover-location` property provides more control over the failover process and allows the failover process to jump to a specified location. Care must be used so that circular failover reference does not take place. Here is an example configuration:

```
austin1.server.com:1389 location=austin preferred-failover-location=boston
london1.server.com:2389 location=london preferred-failover-location=austin
```

```
boston1.server.com:3389 location=boston preferred-failover-location=london
austin2.server.com:4389 location=austin preferred-failover-location=boston
boston2.server.com:5389 location=boston preferred-failover-location=austin
london2.server.com:6389 location=london preferred-failover-location=london
```

The PingDataSync Server will respect the `preferred-failover-location` if it cannot find any external servers in the same location as itself, it will look for any external servers in its own `preferred-failover-location`. In this example, when austin1 becomes unavailable, it will fail over to austin2 because they are in the same location. If austin2 is unavailable, it will fail over to boston1, which is in the `preferred-failover-location` of the PingDataSync Server. If boston1 in unavailable, the PingDataSync Server will fail over to boston2, and finally, it will try the london1 and london2 servers.

Configuration properties that control failover behavior

There are four important advanced properties to fine tune the failover mechanism:

- `max-operation-attempts` (Sync Pipe)
- `response-timeout` (source and destination endpoints)
- `max-failover-error-code-frequency` (source and destination endpoints)
- `max-backtrack-replication-latency` (source endpoints only)

These properties apply to the following LDAP error codes:

LDAP Error Codes

Error Code	Description
ADMIN_LIMIT_EXCEEDED (11)	Indicates that processing on the requested operation could not continue, because an administrative limit was exceeded.
ALIAS_DEREFERENCING_ PROBLEM (36)	Indicates that a problem was encountered while attempting to dereference an alias for a search operation.
CANCELED (118)	Indicates that a cancel request was successful, or that the specified operation was canceled.
CLIENT_SIDE_LOCAL_ERROR (82)	Indicates that a local (client-side) error occurred.
CLIENT_SIDE_ENCODING_ERROR (83)	Indicates that an error occurred while encoding a request.
CLIENT_SIDE_DECODING_ERROR (84)	Indicates that an error occurred while decoding a request.
CLIENT_SIDE_TIMEOUT (85)	Indicates that a client-side timeout occurred.
CLIENT_SIDE_USER_CANCELLED (88)	Indicates that a user canceled a client-side operation.
CLIENT_SIDE_NO_MEMORY (90)	Indicates that the client could not obtain enough memory to perform the requested operation.
CLIENT_SIDE_CLIENT_LOOP (96)	Indicates that a referral loop is detected.
CLIENT_SIDE_REFERRAL_LIMIT_ EXCEEDED (97)	Indicates that the referral hop limit was exceeded.
DECODING_ERROR (84)	Indicates that an error occurred while decoding a

LDAP Error Codes

Error Code	Description
	response.
ENCODING_ERROR (83)	Indicates that an error occurred while encoding a response.
INTERACTIVE_TRANSACTION_ ABORTED (30221001)	Indicates that an interactive transaction was aborted.
LOCAL_ERROR (82)	Indicates that a local error occurred.
LOOP_DETECT (54)	Indicates that a referral or chaining loop was detected while processing a request.
NO_MEMORY (90)	Indicates that not enough memory could be obtained to perform the requested operation.
OPERATIONS_ERROR (1)	Indicates that an internal error prevented the operation from being processed properly.
OTHER (80)	Indicates that an error occurred that does not fall into any of the other categories.
PROTOCOL_ERROR (2)	Indicates that the client sent a malformed or illegal request to the server.
TIME_LIMIT_EXCEEDED (3)	Indicates that a time limit was exceeded while attempting to process the request.
TIMEOUT (85)	Indicates that a timeout occurred.
UNWILLING_TO_PERFORM (53)	Indicates that the server is unwilling to perform the requested operation.

The max-operation-attempts property

The `max-operation-attempts` property (part of the Sync Pipe configuration) specifies the maximum number of times to retry a synchronization operation that fails for reasons other than the Sync Destination being busy, unavailable, server connection closed, or connect error.

To change the default number of retries, use `dsconfig` in non-interactive mode to change the `max-operation-attempts` value on the Sync Pipe object. The following command changes the number of maximum attempts from five (default) to four.

```
$ bin/dsconfig set-sync-pipe-prop \
  --pipe-name "Test Sync Pipe" \
  --set max-operation-attempts:4
```

The response-timeout property

The `response-timeout` property specifies how long the PingDataSync Server should wait for a response from a search request to a source server before failing with LDAP result code 85 (client-side timeout). When a client-side timeout occurs, the Sync Source will retry the request according to the `max-failover-error-code-frequency` property before failing over to a different source server and performing the retry. The total number of retries will not exceed the `max-operation-attempts` property defined in the Sync Pipe configuration. A value of zero indicates that there should be no client-side timeout. The default value is one minute.

Assuming a bidirectional topology, the property can be set with `dsconfig` on the Sync Source and Sync Destination, respectively.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name src \
  --set "response-timeout:8 s"
```

```
$ bin/dsconfig set-sync-destination-prop \
  --destination-name U4389 \
  --set "responsetimeout:9 s"
```

The max-failover-error-code-frequency property

The `max-failover-error-code-frequency` property (part of the Sync Source configuration) specifies the maximum time period that an error code can re-appear until it fails over to another server instance. This property allows the retry logic to be tuned, so that retries can be performed once on the same server before giving up and trying another server. The value can be set to zero if there is no acceptable error code frequency and failover should happen immediately. It can also be set to a very small value (such as 10 ms) if a high frequency of error codes is tolerable. The default value is three minutes.

To change the `max-failover-error-code-frequency` property, use `dsconfig` in non-interactive mode to change the property on the Sync Source object. The following command changes the frequency from three minutes to two minutes.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name source1 \
  --set "max-failover-error-code-frequency:2 m"
```

The max-backtrack-replication-latency property

The `max-backtrack-replication-latency` property (part of the Sync Source configuration) sets the time period that an PingDataSync Server will look for missed changes in the change log due to replication delays. The property should be set to a conservative upper-bound of the maximum replication delay between two servers in the topology. A value of zero implies that there is no limit on the replication latency. The default value is two hours. ThePingDataSync Server stops looking in the change log once it finds a change that is older than the maximum replication latency than the last change it processed on the failed server.

For example, after failing over to another server, the PingDataSync Server must look through the new server’s change log to find the equivalent place to begin synchronizing changes. Normally, the PingDataSync Server can successfully backtrack with only a few queries of the directory, but in some situations, it might have to look further back through the change log to make sure that no changes were missed. Because the changes can come from a variety of sources (replication, synchronization, and over LDAP), the replicated changes between directory servers are interleaved in each change log. When the PingDataSync Server fails over between servers, it has to backtrack to figure out where synchronization can safely pick up the latest changes.

Backtracking occurs until the following:

- The server determines that there is no previous change log state available for any source servers, so it must start at the beginning of the change log.
- The server finds the last processed replication change sequence number (CSN) from the last time it was connected to that replica, if at all. This process is similar to the `set-startpoint` functionality on the `realtime-sync` tool.
- The server finds the last processed replication CSN from every replica that has produced a change so far, and it determines that each change entry in the next oldest batch of changes has already been processed.
- The server finds a change that is separated by more than a certain duration (specified by the `max-backtrack-replication-latency` property) from the most recently processed change.

The following command changes the maximum backtracking from two hours to three hours.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name source1 \
  --set "max-backtrack-replication-latency:3 h"
```

Configure traffic through a load balancer

If a PingData server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request. If the actual client's IP address should be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and the PingData server. See the product documentation for the device type. For PingData servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP) and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

Configure authentication with a SASL external certificate

By default, the PingDataSync Server authenticates to the Directory Server using LDAP simple authentication (with a bind DN and a password). However, the PingDataSync Server can be configured to use SASL EXTERNAL to authenticate to the Directory Server with a client certificate.

> **Note**
> This procedure assumes that PingDataSync Server instances are installed and configured to communicate with the backend Directory Server instances using either SSL or StartTLS.

After the servers are configured, perform the following steps to configure SASL EXTERNAL authentication:

1. Create a JKS keystore that includes a public and private key pair for a certificate that the PingDataSync Server instance(s) will use to authenticate to the Directory Server instance(s). Run the following command in the instance root of one of the PingDataSync Server instances. When prompted for a keystore password, enter a strong password to protect the certificate. When prompted for the key password, press **ENTER** to use the keystore password to protect the private key:

```
$ keytool -genkeypair \
  -keystore config/sync-user-keystore \
  -storetype JKS \
  -keyalg RSA \
  -keysize 2048 \
  -alias sync-user-cert \
  -dname "cn=Sync User,cn=Root DNs,cn=config" \
  -validity 7300
```

2. Create a `config/sync-user-keystore.pin` file that contains a single line that is the keystore password provided in the previous step.
3. If there are other PingDataSync Server instances in the topology, copy the `sync-user-keystore` and `sync-user-keystore.pin` files into the config directory for all instances.
4. Use the following command to export the public component of the user certificate to a text file:

```
$ keytool -export \
  -keystore config/sync-user-keystore \
  -alias sync-user-cert \
  -file config/sync-user-cert.txt
```

5. Copy the `sync-user-cert.txt` file into the `config` directory of all Directory Server instances. Import that certificate into each server's primary trust store by running the following command from the server root. When prompted for the keystore password,

enter the password contained in the `config/truststore.pin` file. When prompted to trust the certificate, enter **yes**.

```
$ keytool -import \
  -keystore config/truststore \
  -alias sync-user-cert \
  -file config/sync-user-cert.txt
```

6. Update the configuration for each PingDataSync Server instance to create a new key manager provider that will obtain its certificate from the `config/sync-user-keystore` file. Run the following `dsconfig` command from the server root:

```
$ dsconfig create-key-manager-provider \
  --provider-name "Sync User Certificate" \
  --type file-based \
  --set enabled:true \
  --set key-store-file:config/sync-user-keystore \
  --set key-store-type:JKS \
  --set key-store-pin-file:config/sync-user-keystore.pin
```

7. Update the configuration for each LDAP external server in each PingDataSync Server instance to use the newly-created key manager provider, and also to use SASL EXTERNAL authentication instead of LDAP simple authentication. Run the following `dsconfig` command:

```
$ dsconfig set-external-server-prop \
  --server-name ds1.example.com:636 \
  --set authentication-method:external \
  --set "key-manager-provider:Sync User Certificate"
```

After these changes, the PingDataSync Server should re-establish connections to the LDAP external server and authenticate with SASL EXTERNAL. Verify that the PingDataSync Server is still able to communicate with all backend servers by running the `bin/status` command. All of the servers listed in the "--- LDAP External Servers ---" section should have a status of `Available`. Review the Directory Server access log can to make sure that the BIND RESULT log messages used to authenticate the connections from the PingDataSync Server include `authType="SASL", saslMechanism="EXTERNAL", resultCode=0`, and `authDN="cn=Sync User,cn=Root DNs,cn=config"`.

Server SDK extensions

Custom server extensions can be created with the Server SDK. Extension bundles are installed from a .zip archive or a file system directory. Use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note
The `manage-extension` tool must be used with Java extensions packaged using the extension bundle

format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation.

The Server SDK enables creating extensions for all PingData servers. Cross-product extensions include:

- Access Loggers
- Alert Handlers
- Error Loggers
- Key Manager Providers
- Monitor Providers
- Trust Manager Providers
- OAuth Token Handlers
- Manage Extension Plugins

Chapter 4: Synchronizing with Active Directory systems

The PingDataSync Server supports full synchronization for newly created or modified accounts with native password changes between directory server, relational databases, and Microsoft Active Directory systems.

This chapter presents configuration procedures for synchronization between Directory Server, Alcatel-Lucent 8661 Directory Server, or other LDAP source servers or targets with Microsoft Active Directory systems.

Topics include:

Overview of configuration tasks

To configure synchronization with Active Directory systems, the following tasks are performed:

- **Enable SSL connections** – If synchronizing passwords between systems, synchronization with Microsoft Active Directory systems requires that SSL be enabled on the Active Directory domain controller, so that the PingDataSync Server can securely propagate the `cn=Sync User` account password and other user passwords to the target.
- **Run the create-sync-pipe-config tool** – On the PingDataSync Server, use the `create-sync-pipe-config` tool to configure the Sync Pipes to communicate with the Active Directory source or target.
- **Configure outbound password synchronization on an Directory Server Sync Source** – After running the `create-sync-pipe-config` tool, determine if outbound password synchronization from an Directory Server Sync Source is required. If so, enable the Password Encryption component on all Directory Server sources that receive password modifications. The Directory Server uses the Password Encryption component, analogous to the Password Sync Agent component, to intercept password modifications and add an encrypted attribute, `ds-changelog-encrypted-password`, to the changelog entry. The component enables passwords to be synchronized securely to the Active Directory system, which uses a different password storage scheme. The encrypted attribute appears in the change log and is synchronized to the other servers, but does not appear in the entries.
- **Configure outbound password synchronization on an Active Directory Sync Source** – After running the `create-sync-pipe-config` tool, determine if outbound password synchronization from an Active Directory Sync Source is required. If so, install the Password Sync Agent (PSA) after configuring the PingDataSync Server.
- **Run the realtime-sync set-startpoint tool** – The `realtime-sync set-startpoint` command may take several minutes to run, because it must issue repeated searches of the Active Directory domain controller until it has paged through all the changes and receives a cookie that is up-to-date.

Configuring synchronization with Active Directory

The following procedure configures a one-way Sync Pipe with the Active Directory topology as the Sync Source and an Directory Server topology as the Sync Destination.

1. From the server-root directory, start the PingDataSync Server.

```
$ <server-root>/bin/start-server
```

2. Run the `create-sync-pipe-config` tool to set up the initial synchronization topology.

   ```
   $ bin/create-sync-pipe-config
   ```

3. On the Initial Synchronization Configuration Tool menu, press **Enter** to continue the configuration.
4. On the Synchronization Mode menu, press **Enter** to select Standard mode.
5. On the Synchronization Directory menu, select the option for one-way (1) or bidirectional (2) for the synchronization topology.
6. On the Source Endpoint Type menu, enter the option for Microsoft Active Directory.
7. On the Source Endpoint Name menu, type a name for the source server, or accept the default.
8. On the Server Security menu, select the security connection type for the source server.
9. On the Servers menu, enter the host name and listener port for the Source Server, or press **Enter** to accept the default (port 636). The server will attempt a connection to the server. After adding the first server, add additional servers for the source endpoints, which will be prioritized below the first server.
10. On the Synchronization User Account DN menu, enter the User Account DN for the source servers. The account will be used exclusively by the PingDataSync Server to communicate with the source external servers. Enter a User Account DN and password. The User Account DN password must meet the minimum password requirements for Active Directory domains.
11. Set up the Destination Endpoint servers.

The Active Directory Sync User account

The Sync User created for Active Directory is added to the `cn=Administrators` branch and is given most of a root user's permissions. If this account cannot be secured and there is a need to configure the permissions required by the Sync User, the following are required to perform synchronization tasks:

As a Sync Source, these permissions are needed:

- List contents
- Read all properties
- Read permissions

Deleted items are a special case. For the Sync Server to see deleted entries, the user account must have sufficient access to `cn=Deleted Objects,<domain name>`. Giving access to that DN requires using the `dsacls` tool, such as:

```
# Take ownership may be required to make the needed changes.
dsacls "CN=Deleted Objects,DC=example,DC=com" /takeOwnership
```

```
# Give the Sync User generic read permission to the domain.
dsacls "CN=Deleted Objects,DC=example,DC=com" /G "example\SyncUser":GR
```

```
# List the permisison for the domain.
dsacls "CN=Deleted Objects,DC=example,DC=com"
```

To revoke all permissions from the Sync User, run the following `dsacls` command:

```
dsacls "CN=Deleted Objects,DC=example,DC=com" /R "example\SyncUser"
```

If Active Directory is used as a destination for synchronization, the Sync User account should not be changed.

Prepare external servers

Perform the following steps to prepare external servers:

1. After configuring the source and destination endpoints, the PingDataSync Server prompts to "prepare" each external server. The process requires trusting the certificate presented to the server, and then testing the connection. If this step is not performed, the process can be completed after configuring the Sync Pipes using the `prepare-endpoint-server` tool.
2. Configuring this server for synchronization requires manager access. Enter the DN and password of an account capable of managing the external directory server.
3. Enter the maximum age of changelog entries. The value is formatted as `[number][time-unit]`, where the time unit format resembles ("8h" for eight hours, "3d" for three days, "1w" for one week). Setting this value caps how long the PingDataSync Server can be offline. A smaller value limits how many changes are stored and is necessary to limit excessive changelog growth in high-modification environments.
4. To prepare another server in the topology, follow the prompts. The previously entered manager credentials can be reused to access additional servers. Repeat the process for each server configured in the system.

Configure Sync Pipes and Sync Classes

Perform the following steps to configure Sync Pipes and Sync Classes:

1. On the Sync Pipe Name menu, type a unique name to identify the Sync Pipe, or accept the default.
2. On the Pre-Configured Sync Class Configuration for Active Directory Sync Source menu, enter **yes** to synchronize user CREATE operations, and enter the object class for the user entries at the destination server, or accept the default (`user`). To synchronize user MODIFY and DELETE operations from Active Directory, enter **yes**.

3. To synchronize passwords from Active Directory, press **Enter** to accept the default (yes). If synchronizing passwords from Active Directory, install the Ping Identity Password Sync Agent component on each domain controller.
4. To create a DN map for the user entries in the Sync Pipe, enter the base DN for the user entries at the Microsoft Active Directory Sync Source, then enter the base DN for the user entries at the PingDataSync Server Destination.
5. A list of basic attribute mappings from the Microsoft Active Directory Source to the Directory Server destination is displayed. More complex attribute mappings involving constructed or DN attribute mappings must be configured with the `dsconfig` tool. The following is a sample mapping.

```
Below is a list of the basic mappings that have been set up for user
entries synchronized from Microsoft Active Directory -> Directory Server.
You can add to or modify this list with any direct attribute mappings. To
set up more complex mappings (such as constructed or DN attribute
mappings), use the 'dsconfig' tool.

1) cn -> cn
2) sn -> sn
3) givenName -> givenName
4) description -> description
5) sAMAccountName -> uid
6) unicodePwd -> userPassword
```

6. Enter the option to add a new attribute mapping. Enter the source attribute, and then enter the destination attribute. The following example maps the `telephoneNumber` attribute (Active Directory) to the `otherTelephone` attribute (Directory Server).

```
Select an attribute mapping to remove, or choose 'n' to add a new one
[Press ENTER to continue]: n

Enter the name of the source attribute: telephoneNumber
Enter the name of the destination attribute: otherTelephone
```

7. If synchronizing group CREATE, MODIFY, and DELETE operations from Active Directory, enter **yes**.
8. Review the basic user group mappings.
9. On the Sync Pipe Sync Class Definitions menu, enter another name for a new Sync Class if required. Repeat steps 2–7 to define this new Sync Class. If no additional Sync Class definitions are required, press **Enter** to continue.
10. Review the Sync Pipe Configuration Summary, and accept the default ("write configuration"), which records the commands in a batch file (`sync-pipe-cfg.txt`). The batch file can be used to set up other topologies. The following summary shows two Sync Pipes and its associated Sync Classes.

```
>>>> Configuration Summary

  Sync Pipe: AD to Directory Server
    Source: Microsoft Active Directory
      Type: Microsoft Active Directory
      Access Account: cn=Sync
User,cn=Users,DC=adsync,DC=PingIdentity,DC=com
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Servers: 10.5.1.149:636

    Destination: Directory Server
      Type: Directory Server
      Access Account: cn=Sync User,cn=Root DNs,cn=config
      Base DN: dc=example,dc=com
      Servers: localhost:389

    Sync Classes:
      Microsoft Active Directory Users Sync Class
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Filters: (objectClass=user)
      DN Map: **,CN=Users,DC=adsync,DC=PingIdentity,DC=com ->{1},ou=users,
      dc=example,dc=com
      Synchronized Attributes: Custom set of mappings are defined
      Operations: Creates,Deletes,Modifies

  Sync Pipe: Directory Server to AD
    Source: Directory Server
      Type: Directory Server
      Access Account: cn=Sync User,cn=Root DNs,cn=config
      Base DN: dc=example,dc=com
      Servers: localhost:389

    Destination: Microsoft Active Directory
      Type: Microsoft Active Directory
      Access Account: cn=Sync
User,cn=Users,DC=adsync,DC=PingIdentity,DC=com
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Servers: 10.5.1.149:636

    Sync Classes:
       Directory Server Users Sync Class
      Base DN: dc=example,dc=com
      Filters: (objectClass=inetOrgPerson)
      DN Map: **,ou=users,dc=example,dc=com ->{1},CN=Users,DC=adsync,
       DC=PingIdentity,DC=com
      Synchronized Attributes: Custom set of mappings are defined
      Operations: Creates,Deletes,Modifies
```

11. To apply the configuration to the local PingDataSync Server instance, type **yes**. The configuration is recorded at `<server-root>/logs/tools/createsync-pipe-`

config.log.

Configure password encryption

This procedure is required if synchronizing passwords from an Directory Server to Active Directory, or if synchronizing clear text passwords. These steps are not required if synchronizing from Active Directory to an PingData Directory Server, or if not synchronizing passwords.

1. On the Directory Server that will receive the password modifications, enable the Change Log Password Encryption component. The component intercepts password modifications, encrypts the password and adds an encrypted attribute, ds-changelog-encrypted-password, to the change log entry. The encryption key can be copied from the output if displayed, or accessed from the <serverroot>/bin/sync-pipe-cfg.txt file.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Changelog Password
Encryption" \
  --set enabled:true \
  --set changelog-password-encryption-key:<key>
```

2. On the PingDataSync Server, set the decryption key used to decrypt the user password value in the change log entries. The key allows the user password to be synchronized to other servers that do not use the same password storage scheme.

```
$ bin/dsconfig set-global-sync-configuration-prop \
  --set changelog-password-decryption-key:ej5u9e39pqo68
```

Test the configuration or populate data in the destination servers using bulk resync mode. See Using the resync Tool on the Identity Sync Server. Then, use realtime-sync to start synchronizing the data. See Using the realtime-sync Tool for more information. If synchronizing passwords, install the Password Sync Agent (PSA) on all of the domain controllers in the topology.

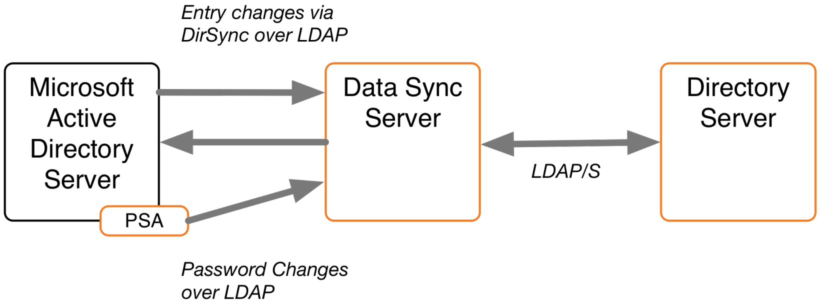
The Password Sync Agent

When synchronizing passwords with Active Directory systems, the PingDataSync Server requires that the Ping Identity Password Sync Agent (PSA) be installed on all domain controllers in the synchronization topology. This component provides real-time outbound password synchronization from Microsoft Active Directory to any supported Sync Destinations.

The PSA component provides password synchronization between directories that support differing password storage schemes. The PSA immediately hashes the password with a 160-bit salted secure hash algorithm and erases the memory where the clear-text password was stored. The component only transmits data over a secure (SSL) connection, and follows Microsoft's security guidelines when handling clear-text passwords. The PSA also uses Microsoft Windows password filters, which are part of the local security authority (LSA) process. The password filters enable implementing password policy validation and change notification mechanisms. For more information, see Microsoft's product documentation.

Note

For outbound password synchronization from an Directory Server to Active Directory, enable the Password Encryption component. See Configuring the Password Encryption Component for more information.



Password Synchronization with Microsoft Active Directory

The PSA supports failover between servers. It caches the hashed password changes in a local database until it can be guaranteed that all PingDataSync Servers in the topology have received them. The failover features enable any or all of the PingDataSync Servers to be taken offline without losing any password changes from Active Directory.

The PSA is safe to leave running on a domain controller indefinitely. To stop synchronizing passwords, remove the `userPassword` attribute mapping in the PingDataSync Server, or stop the server. The PSA will not allow its local cache of password changes to grow out of control; it automatically cleans out records from its local database as soon as they have been acknowledged. It also purges changes that have been in the database for more than a week.

Before installing the PSA, consider the following:

- Make sure that the Active Directory domain controller has SSL enabled and running.
- Make sure the PingDataSync Servers are configured to accept SSL connections when communicating with the Active Directory host.
- At least one Active Directory Sync Source (ADSyncSource) needs to be configured on the PingDataSync Server and should point to the domain controller(s) on which the PSA will reside.
- At the time of installation, all PingDataSync Servers in the sync topology must be online and available.
- The PSA component is for outbound-only password synchronization from the Active Directory Systems. It is not necessary if performing a one-way password synchronization from the Directory Server to the Active Directory server.

Install the Password Sync Agent

The Directory Server distributes the PSA in zip file format with each PingDataSync Server package. The initial installation of the PSA requires a system restart.

Perform the following steps to install the PSA

1. On the domain controller, double-click the `setup.exe` file to start the installation.
2. Select a folder for the PSA binaries, local database, and log files.
3. Enter the host names (or IP addresses) and SSL ports of the PingDataSync Servers, such as `sync.host.com:636`. Do not add any prefixes to the hostnames.
4. Enter the Directory Manager DN and password. This creates an ADSync user on the PingDataSync Server.
5. Enter a secret key (password) that the sync agent uses when the ADSync user connects to the PingDataSync Server instances.
6. Click **Next** to begin the installation. All of the specified PingDataSync Servers are contacted, and any failures will roll back the installation. If everything succeeds, a message displays indicating that a restart is required. The PSA will start when the computer restarts, and the LSA process is loaded into memory. The LSA process cannot be restarted at runtime.
7. If synchronizing pre-encoded passwords from Active Directory to a Ping Identity system, allow pre-encoded passwords in the default password policy.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set allow-pre-encoded-passwords:true
```

Upgrade or Uninstall the Password Agent

The PingDataSync Server provides the `update` tool for upgrades to the server code, including the PSA. The upgrade does not require a restart, because the core password filter is already running under LSA. The upgrade replaces the implementation binaries, which are encapsulated from the password filter DLL.

To uninstall the PSA on the Active Directory system, use **Add/Remove Programs** on the Windows Control Panel. The implementation DLL will be unloaded, and the database and log files are deleted. Only the binaries remain.

The core password filter will still be running under the LSA process. It imposes zero overhead on the domain controller, because the implementation DLL has been unloaded. To remove the password filter itself (located at `C:\WINDOWS\System32\ubidPassFilt.dll`), restart the computer. On restart, the password filter and implementation binaries (in the install folder) can be deleted.

Note

The PSA cannot be reinstalled without another reboot.

Manually Configure the Password Sync Agent

Configuration settings for the Password Sync Service are stored in the Windows registry in `HKLM\SOFTWARE\UnboundID\PasswordSync`. Configuration values under this registry key can be modified during runtime. The agent automatically reloads and refresh its settings from the registry. Verify that the agent is working by checking the current log file, located in `<server-root>\logs\password-sync-current.log`.

Chapter 5: Synchronize with relational databases

The PingDataSync Server supports high-scale, highly-available data synchronization between the directory servers and relational database management systems (RDBMS). Any database with a JDBC 3.0 or later driver can be used.

Topics include:

Use the Server SDK

Synchronizing LDAP data to or from a relational database requires creating a JDBC Sync Source or Destination extension to act as an interface between the PingDataSync Server and the relational database. The Server SDK provides APIs to develop plug-ins and third-party extensions to the server using Java or Groovy. The Server SDK's documentation is delivered with the Server SDK build in zip format.

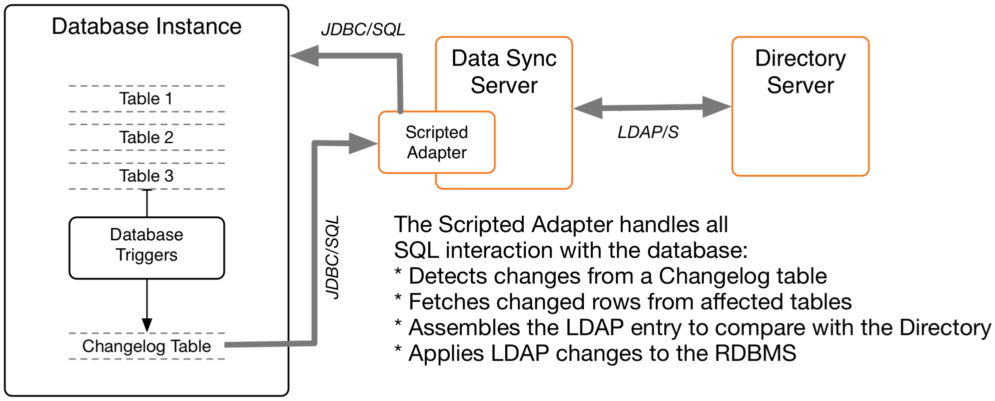
> **Note**
> Server SDK support is provided with Premium Support for the each product. Ping Identity does not provide support for the third party extensions developed using the Server SDK. Contact a Ping Identity support representative for assistance.

The Server SDK contains two abstract classes that correspond to how the database is used:

- `com.unboundid.directory.sdk.sync.api.JDBCSyncSource`
- `com.unboundid.directory.sdk.sync.api.JDBCSyncDestination`

The remainder of the SDK contains helper classes and utility functions to facilitate the script implementation. The SDK can use any change tracking mechanism to detect changes in the database. Examples are provided in the `<server-root>/config/jdbc/samples` directory for Oracle Database and Microsoft SQL Server.

The PingDataSync Server uses a scripted adapter layer to convert any database change to an equivalent LDAP entry. The Sync Pipe then processes the data through inclusive (or exclusive) filtering using attribute and DN maps defined in the Sync Classes to update the endpoint servers. For example, a script using Java can be configured by setting the `extension-class` property on a `ThirdPartyJDBCSyncSource` or `ThirdPartyJDBCSyncDestination` configuration object within the PingDataSync Server. The following is a sample architecture.



The Scripted Adapter handles all
SQL interaction with the database:
* Detects changes from a Changelog table
* Fetches changed rows from affected tables
* Assembles the LDAP entry to compare with the Directory
* Applies LDAP changes to the RDBMS

Synchronizing with RDBMS Overview

The RDBMS synchronization process

The PingDataSync Server synchronizes data between a directory server and an RDBMS system with a Server SDK extension. The PingDataSync Server provides multiple configuration options, such as advanced filtering (fractional and subtree), attribute and DN mappings, transformations, correlations, and configurable logging.

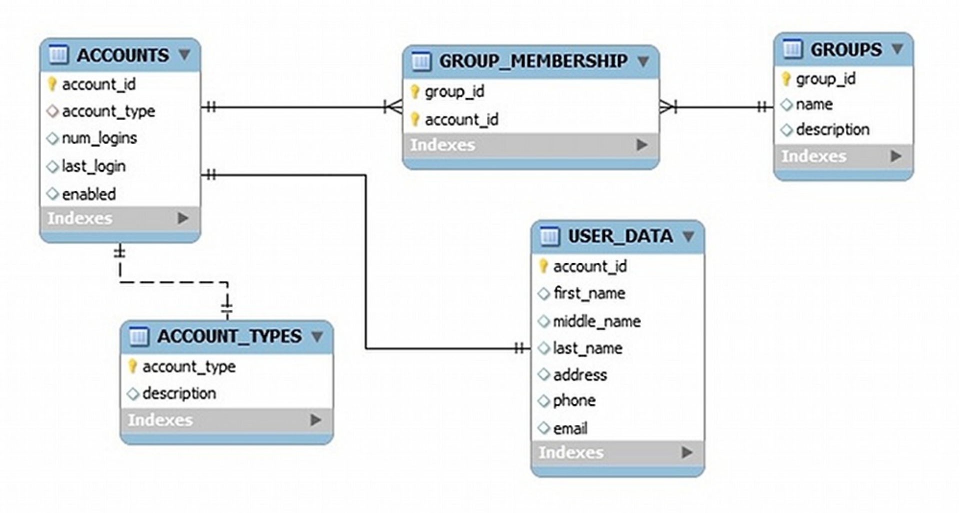
To support synchronizing changes, the database must be configured with a change tracking mechanism. An approach involving triggers, (one trigger per table) to record all changes to a change log table, is recommended. The database change log table Ping Identityshould record the type of change (INSERT, UPDATE, DELETE), the specific table name, the unique identifier for the changed row, the database entry type, the changed columns (from the source table), the modifier's name, and the change timestamp.

The PingDataSync Server delegates the physical interaction with the database to a user-defined extension, which has full control of the SQL queries. The extension layer provides flexibility in how the mapping semantics between the LDAP environment and the relational database environment are defined. The connection management, pooling, retry logic, and other boilerplate code are handled internally by the PingDataSync Server.

The RDBMS Synchronization (DBSync) implementation does not support failover between different physical database servers. Most enterprise databases have a built-in failover layer from which the PingDataSync Server can point to a single virtual address and port and still be highly available. A single RDBMS node can scale to multiple directory server endpoints.

DBSync example

ThePingDataSync Server provides a DBSync example between two endpoints consisting of an Ping Identity Directory Server source and a RDBMS system, which will be used in this chapter. The entity-relational diagram for the normalized database schema is available in `<server-root>/config/jdbc/samples/oracle-db/ComplexSchema.jpg`, and is illustrated here:



Entry Relation Diagrams for the Schema Tables

Five tables are represented with their primary keys in bold. The entity relations and foreign keys are marked by the relationship lines. The illustration shows mapping to a custom object class on the directory server, while the "groups" table maps to a standard LDAP group entry with `objectclass:groupOfUniqueNames`.

Example directory server entries

The following example assumes that the directory server's schema is configured to handle the mapped attributes. If configuring a database-to-directory Sync Pipe with a newly installed directory server, make sure that the schema has the correct `attributeType` and `objectClass` definitions in place. The definitions can be added in a custom `99-user.ldif` file in the `config/schema` folder of the directory server, if necessary. The following are the LDAP entries that are used in the synchronization example:

```
dn: accountid=0,ou=People,dc=example,dc=com
objectClass: site-user
firstName: John
lastName: Smith
accountID: 1234
email: jsmith@example.com
phone: +1 556 805 4454
address: 17121 Green Street
numLogins: 4
lastLogin: 20070408182813.196Z
enabled: true

dn: cn=Group 1,ou=Groups,dc=example,dc=com
objectClass: groupOfUniqueNames
description: This is Group 1
uniqueMember: accountID=0,ou=People,dc=example,dc=com
uniqueMember: accountID=1,ou=People,dc=example,dc=com
```

Configure DBSync

Configuring a DBSync system includes extra tasks to create the extensions and to configure the database. The overall configuration process is as follows:

1. Download the appropriate JDBC driver to the PingDataSync Server's `/PingDataSync/lib` directory, and restart the server for the driver to load into the runtime.
2. Open the `java.properties` file with a text editor and add the `jdbc.drivers` argument. Save the file.
3. Run the `dsjavaproperties` command to apply the change. For example, enter the following for `start-sync-server`:

   ```
   start-sync-server.java-args=-d64 -server -Xmx256m -Xms256m -
   XX:+UseConcMarkSweepGC -
   ```

```
Djdbc.drivers=foo.bah.Driver:wombat.sql.Driver:com.example.OurDriver ...
etc.
```

4. Create one or more JDBC extensions based on the Server SDK. If configuring for bidirectional synchronization, two scripts are needed: one for the JDBC Sync Source; the other for the JDBC Sync Destination. Place the compiled extensions in the `/lib/extensions` directory.
5. Configure the database change log table and triggers (presented later). The vendor's native change tracking mechanism can be used, but a change log table should also be configured. Each table requires one database trigger to detect the changes and loads them into the change log table.
6. Configure the Sync Pipes, Sync Classes, external servers, DN and attribute maps for one direction.
7. Run the `resync --dry-run` command to test the configuration settings.
8. Run `realtime-sync set-startpoint` to initialize the starting point for synchronization.
9. Run the `resync` command to populate data on the destination endpoint.
10. Start the Sync Pipes using the `realtime-sync start` command.
11. Monitor the PingDataSync Server using the `status` commands and logs.
12. For bidirectional synchronization, configure another Sync Pipe, and repeat steps 4–8 to test the system.

Create the JDBC extension

The JDBC extension implementation must be written in Java, or the Groovy scripting language. Consult the Server SDK documentation for details on how to build and deploy extensions. The examples in this guide use Java. Java extensions are more strict and will catch programming errors during compile time rather than at runtime. Groovy is more flexible and can accomplish more with less lines of code.

Groovy scripts must reside in the `/lib/groovy-scripted-extensions` directory (Java implementations reside in `/lib/extensions`), which may also contain other plug-ins built using the Server SDK. If a script declares a package name, it must live under the corresponding folder hierarchy, just like a Java class. For example, to use a script class called `ComplexJDBCSyncSource` whose package is `com.unboundid.examples.oracle`, place it in `/lib/groovy-scripted-extensions/com/unboundid/examples/oracle` and set the `script-class` property on the Sync Source to `com.unboundid.examples.oracle.ComplexJDBCSyncSource`. There are a few reference implementations provided in the `config/jdbc/samples` directory. Use the `manage-extension` tool in the `bin` directory, or `bat` (Windows) to install or update the extension. See the Server SDK Extensions section for more information.

Note
Any changes to an existing script require a manual Sync Pipe restart. Any configuration change automatically restarts the affected Sync Pipe.

The default libraries available on the classpath to the script implementation include:

- Groovy
- LDAP SDK for Java
- JRE

Logging from within a script can be done with the Server SDK's `ServerContext` abstract class. Some of `ServerContext` methods are not available when the `resync` tool runs, because it runs outside of the PingDataSync Server Server process. Any logging during a `resync` operation is saved to the `logs/tools/resync.log` file.

Implement a JDBC Sync Source

The `JDBCSyncSource` abstract class must be implemented to synchronize data from a relational database. Since the PingDataSync Server is LDAP-centric, this class enables database content to be converted into LDAP entries. For more detailed information on the class, see the Server SDK Javadoc.

The extension imports classes from the Java API, LDAP SDK for Java API, and the Server SDK. Depending on the data, implement the following methods:

- `initializeJDBCSyncSource` – Called when a Sync Pipe first starts, or when the `resync` process starts. Any initialization should be performed here, such as creating internal data structures and setting up variables.
- `finalizeJDBCSyncSource` – Called when a Sync Pipe stops, or when the `resync` process stops. Any clean up should be performed here, and all internal resources should be freed.
- `setStartpoint` – Sets the starting point for synchronization by identifying the starting point in the change log. This method should cause all changes previous to the specified start point to be disregarded and only changes after that point to be returned by the `getNextBatchOfChanges` method. There are several different startpoint types (see SetStartpointOptions in the Server SDK), and this implementation is not required to support them all. If the specified startpoint type is unsupported, this method throws an exception (`IllegalArgumentException`). This method can be called from two different contexts: when the `realtime-sync set-startpoint` command is used (the Sync Pipe is required to be stopped) or immediately after a connection is established to the source server.

Note
The `RESUME_AT_SERIALIZABLE` startpoint type must be supported. This method is used when a Sync Pipe first starts and loads its state from disk.

- getStartpoint – Gets the current value of the startpoint for change detection.
- fetchEntry – Returns a full source entry (in LDAP form) from the database, corresponding to the DatabaseChangeRecord object that is passed. The resync command also uses this class to retrieve entries.
- acknowledgeCompletedOps – Provides a means for the PingDataSync Server to acknowledge to the database which operations have completed processing.

Note
The internal value for the startpoint should only be updated after a synchronization operation is acknowledged in to this script (through this method). If not, changes could be missed when the PingDataSync Server is restarted.

- getNextBatchOfChanges – Retrieves the next set of changes for processing. The method also provides a generic means to limit the size of the result set.
- listAllEntries – Used by the resync command to get a listing of all entries.
- cleanupChangelog – In general, we recommend implementing a cleanupChangelog method, so that the PingDataSync Server can purge old records from the change log table, based on a configurable age.

See the config/jdbc/samples directory for example script implementations and the Server SDK Javadoc for more detailed information on each method.

Implement a JDBC Sync Destination

The JDBCSyncDestination abstract class must be implemented to synchronize data into a relational database. The class enables converting LDAP content to database content. The extension imports classes from the Java API, LDAP SDK for Java API, and the Server SDK, depending on the database configuration. Implement the following methods in the script:

- initializeJDBCSyncDestination – Called when a Sync Pipe starts, or when the resync process starts. Any initialization should be performed here, such as creating internal data structures and setting up variables.
- finalizeJDBCSyncDestination – Called when a Sync Pipe stops, or when the resync process stops. Any clean up should be performed here, and all internal resources should be freed.
- createEntry – Creates a full database entry (or row), corresponding to the LDAP entry that is passed in.
- modifyEntry – Modifies a database entry, corresponding to the LDAP entry that is passed in.
- fetchEntry – Returns a full destination database entry (in LDAP form), corresponding to the source entry that is passed in.

- `deleteEntry` – Deletes a full entry from the database, corresponding to the LDAP entry that is passed in.

For more detailed information on the abstract class, consult the Server SDK Javadoc.

Configure the database for synchronization

Configuring the database for synchronization includes defining:

- a database SyncUser account
- the change tracking mechanism
- the database triggers (one per table) for the application

The following procedure uses the example setup script in `/config/jdbc/samples/oracle-db/OracleSyncSetup.sql`. Items in brackets are user-named labels.

> **Note**
> Database change tracking necessary if synchronizing FROM the database. If synchronizing TO a database, configure the Sync User account and the correct privileges.

1. Create an Oracle login (`SyncUser`) for the PingDataSync Server, so that it can access the database server. Grant sufficient privileges to the `SyncUser` for any tables to be synchronized, and change the default password.

```
CREATE USER SyncUser IDENTIFIED BY password
DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;
GRANT "RESOURCE" TO SyncUser;
GRANT "CONNECT" TO SyncUser;
```

2. Create change log tables on the database as follows:

```
CREATE TABLE ubid_changelog (
  --This is the unique number for the change change_number Number NOT NULL
PRIMARY KEY,
  --This is the type of change (insert, update, delete). NOTE: This should
represent
  --the actual type of change that needs to happen on the destination(for
example a
  --database delete might translate to a LDAPmodify, etc.)
    change_type VARCHAR2(10) NOT NULL,

  --This is the name of the table that was changed table_name VARCHAR(50)
NOT NULL,
  --This is the unique identifier for the row that was changed. It is up
to
  --the trigger code to construct this, but it should follow a DN-like
format
  --(e.g. accountID={accountID}) where at least the primary key(s) are
  --present. If multiple primary keys are required, they should be
delimited
  --with a unique string, such as '%%' (e.g. accountID={accountID}%%
  --groupID={groupID})
    identifier VARCHAR2(100) NOT NULL,

  --This is the database entry type. The allowable values for this must be
  --set on the JDBC Sync Source configuration within the Synchronization
  --Server.
    entry_type VARCHAR2(50) NOT NULL,

  --This is a comma-separated list of columns from the source table that
were updated as part of
  --this change.
    changed_columns VARCHAR2(1000) NULL,

  --This is the name of the database user who made the change
    modifiers_name VARCHAR2(50) NOT NULL,

  --This is the timestamp of the change
    change_time TIMESTAMP(3) NOT NULL, CONSTRAINT chk_change_type
    CHECK (change_type IN ('insert','update','delete'))) ORGANIZATION
INDEX;
```

3. Create an Oracle function to get the `SyncUser` name. This is a convenience function for the triggers.

```
CREATE OR REPLACE FUNCTION get_sync_user RETURN VARCHAR2
IS
BEGIN
  RETURN 'SyncUser';
END get_sync_user;
```

4. Create an Oracle sequence object for the change-number column in the change log table.

```
CREATE SEQUENCE ubid_changelog_seq MINVALUE 1 START WITH 1
NOMAXVALUE INCREMENT BY 1 CACHE 100 NOCYCLE;
```

5. Create a database trigger for each table that will participate in synchronization. An example, located in `/config/jdbc/samples/oracle-db/OracleSyncSetup.sql`, shows a trigger for the Accounts table that tracks all changed columns after any INSERT, UPDATE, and DELETE operation. The code generates a list of changed items and then inserts them into the change log table.

Considerations for synchronizing to database destination

When configuring a directory-to-database Sync Pipe, the following are recommended:

- **Identify the Object Classes** – Create a Sync Class per object class, so that they can easily be distinguished and have different mappings and synchronization rules.

- For each Sync Class, set the following items in the configuration menus using the `dsconfig` tool.
  - **Set the Include-Filter Property** – Make sure the `include-filter` property is set on the Sync Class configuration menu to something that will uniquely identify the source entries, such as `objectClass=customer`.
  - **Create Specific Attribute Mappings** – Create an attribute mapping for every LDAP attribute to be synchronized to a database column, add these to a single attribute map, and set it on the Sync Class. With this configured, the script does not need to know about the schema on the directory side. A constructed attribute mapping that maps a literal value to the `objectClass` attribute can be added to the script to determine the database entry type. For example, `"account" -> objectClass` can be added, which would result in the constructed destination LDAP entry always containing an `objectClass` of `"account."` If needed, a mulit-valued `conditional-value-pattern` property can be used to conditionalize the attribute mapping based on the subtype of the entry or on the value of the attribute. See Conditional Value Mapping for additional information.
  - **Create Specific DN Maps (optional)** – If necessary, create a DN map that recognizes the DN's of the source entries and maps them to a desired destination DN. In most cases, the script will use the attributes rather than the DN to figure out which database entry needs to be changed.
  - **Set auto-mapped-source-attribute to "-none-"** – Remove the default value of "-all-" from the `auto-mapped-source-attribute` on the Sync Class configuration menu, and replace it with "-none-."
- **Configure Create-Only Attributes** – Any attributes that should be included when created, but never modified (such as `objectclass`) should be specified on the Sync Pipe as a `create-only` attribute. If the PingDataSync Server ever computes a difference in that attribute between the source and destination, it will not try to modify it at the destination. To avoid bidirectional loop-back, set the `ignore-changes-by-[user|dn]` property on both Sync Sources when configuring for bidirectional synchronization.
- **Synchronizing DELETE Operations** – On Directory Server and Alcatel-Lucent 8661 Directory Server systems, configure the `changelog-deleted-entry-include-attribute` property on the changelog backend menu using the `dsconfig` tool. This property allows for the proper synchronization of DELETE operations. For more information, see Configuring the Directory Server Backend for Synchronizing Deletes.
- **Attribute-Synchronization-Mode for DBSync** – For MODIFY operations, the PingDataSync Server detects any change on the source change log, fetches the source entry, applies mappings, computes the equivalent destination entry, fetches the actual destination entry, and then runs a diff between the two entries to determine the minimal set of changes to synchronize. By default, changes on the destination entry are made

only for those attributes that were detected in the original change log entry. This is configurable using the `attribute-synchronization-mode` property, which sets the type of diff operation that is performed between the source and destination entries.

If the source endpoint is a database server, set the `attribute-synchronization-mode` property to `all-attributes` on the Sync Class configuration menu. The diff operation will consider all source attributes. Any that have changed will be updated on the destination, even if the change was not originally detected in the change log. This mode is useful when a list of changed columns in the database may not be available. If both endpoints are directory servers, use the default configuration of `modified-attributes-only` to avoid possible replication conflicts.

- **Handling MODDN Operations** – The concept of renaming an entry (modifyDN) does not have a direct equivalent for relational databases. The `JDBCSyncDestination` API does not handle changes of this type. Instead, the `modifyEntry()` method is called as if it is a normal change. The extension can verify if the entry was renamed by looking at the `SyncOperation` that is passed in (`syncOperation.isModifyDN()`). If true, the `fetchedDestEntry` parameter will have the old DN. The new DN can be obtained by calling `syncOperation.getDestinationEntryAfterChange()`.

Configure a directory-to-database Sync Pipe

The following configures a one-way Sync Pipe with an Directory Server as the Sync Source and an RDBMS (Oracle) system as the Sync Destination with the `create-sync-pipe-config` tool. Sync Pipes can be configured later using `dsconfig`.

Create the Sync Pipe

The following procedures configure the Sync Pipe, external servers, and Sync Classes. The examples are based on the Complex JDBC sample in the `config/jdbc/samples/oracle-db` directory.

The `create-sync-pipe-config` tool can be run with the server offline and the configuration can later be imported.

1. Run the `create-sync-pipe-config` tool.

   ```
   $ bin/create-sync-pipe-config
   ```

2. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.
3. On the Synchronization Mode menu, select Standard Mode or Notification Mode.
4. On the Synchronization Directory menu, choose one-way or bidirectional synchronization.

Configure the Sync Source

1. On the Source Endpoint Type menu, enter the number for the sync source corresponding to the type of source external server.
2. Enter a name for the Source Endpoint.
3. Enter the base DN for the directory server, which is used as the base for LDAP searches. For example, enter `dc=example,dc=com`, and then press **Enter** again to return to the menu. If entering more than one base DN, make sure the DNs do not overlap.
4. On the Server Security menu, select the type of communication that the PingDataSync Server will use with the endpoint servers.
5. Enter the host and port of the source endpoint server. The Sync Source can specify a single server or multiple servers in a replicated topology. The server tests that a connection can be established.
6. Enter the DN of the Sync User account and create a password for this account. The Sync User account enables the PingDataSync Server to access the source endpoint server. By default, the Sync User account is stored as `cn=SyncUser,cn=Root DNs,cn=config`.

Configure the destination endpoint server

1. On the Destination Endpoint Type menu, select the type of data store on the endpoint server. This example is configuring an Oracle Database.
2. Enter a name for the Destination Endpoint.
3. On the JDBC Endpoint Connection Parameters menu, enter the fully-qualified host name or IP address for the Oracle database server.
4. Enter the listener port for the database server, or press **Enter** to accept the default (1521).
5. Enter a database name such as `dbsync-test`.
6. The server attempts to locate the JDBC driver in the `lib` directory. If the file is found, a success message is displayed.

   ```
   Successfully found and loaded JDBC driver for:
   jdbc:oracle:thin:@//dbsync-w2k8-vm-2:1521/dbsync-test
   ```

   If the server cannot find the JDBC driver, add it later, or quit the `create-sync-pipe-config` tool and add the file to the `lib` directory.
7. Add any additional JDBC connection properties for the database server, or press **Enter** to accept the default (no). Consult the JDBC driver's vendor documentation for supported properties.

8. Enter a name for the database user account with which the PingDataSync Server will communicate, or press **Enter** to accept the default (SyncUser). Enter the password for the account.
9. On the Standard Setup menu, enter the number for the language (Java or Groovy) that was used to write the server extension.
10. Enter the fully qualified name of the Server SDK extension class that implements the JDBCSyncDestination API.

    ```
    Enter the fully qualified name of the Java class that will implement
    com.unboundid.directory.sdk.sync.api.JDBCSyncDestination:
    com.unboundid.examples.oracle.ComplexJDBCSyncDestination
    ```

11. Configure any user-defined arguments needed by the server extension. These are defined in the extension itself and the values are specified in the server configuration. If there are user-defined arguments, enter **yes**.
12. To prepare the Source Endpoint server, which tests the connection to the server with the Sync User account, press **Enter** to accept the default (yes). For the Sync User account, it will return "Denied" as the account has not been written yet to the Directory Server at this time.

    ```
    Testing connection to server1.example.com:1389 ..... Done
    Testing 'cn=Sync User,cn=Root DNs,cn=config' access ..... Denied
    ```

13. To configure the Sync User account on the directory server, press **Enter** to accept the default (yes). Enter the bind DN (`cn=Directory Manager`) and the bind DN password of the directory server so that you can configure the `cn=Sync User` account. The PingDataSync Server creates the Sync User account, tests the base DN, and enables the change log.

    ```
    Created 'cn=Sync User,cn=Root DNs,cn=config'
    Verifying base DN 'dc=example,dc=com' ..... Done
    Enabling cn=changelog .....
    ```

14. Enter the maximum age of the change log entries, or press **Enter** to accept the default.

Configure the Sync Pipe and Sync Classes

The following procedures define a Sync Pipe and two Sync Classes. The first Sync Class is used to match the `accounts` objects. The second Sync Class matches the `group` objects.

1. Continuing from the previous session, enter a name for the Sync Pipe.
2. When prompted to define one or more Sync Classes, enter **yes**.

Configure the accounts Sync Class

1. Enter a name for the Sync Class. For example, type `accounts_sync_class`.
2. If restricting entries to specific subtrees, enter one or more base DNs. If not, press **Enter** to accept the default (no).
3. To set an LDAP search filter, type **yes** and enter the filter `"(accountid=*)"`. Press **Enter** again to continue. This property sets the LDAP filters and returns all entries that match the search criteria to be included in the Sync Class. In this example, specify that any entry with an `accountID` attribute be included in the Sync Class. If the entry does not contain any of these values, it will not be synchronized to the target server.
4. Choose to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization, or press **Enter** to accept the default (all).
5. Specify the operations that will be synchronized for the Sync Class, or press **Enter** to accept the default.

Configure the groups Sync Class

For this example, configure another Sync Class to handle the `groups` objectclass. The procedures are similar to that of the configuration steps for the `account_sync_class` Sync Class.

1. On the Sync Class menu, enter a name for a new sync class, such as `groups_sync_class`.
2. To restrict entries to specific subtrees, enter one or more base DNs.
3. Set an LDAP search filter. Type **yes** to set up a filter and enter the filter `"(objectClass=groupOfUniqueNames)."` This property sets the LDAP filters and returns all entries that match the `groupOfUniqueNames` attribute to be included in the Sync Class. If the entry does not contain any of these values, it will not be synchronized to the target server.
4. Choose to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization, or press **Enter** to accept the default (all).
5. Specify the operations that will be synchronized for the Sync Class, or press **Enter** to accept the default.
6. At the prompt to enter the name of another Sync Class, press **Enter** to continue.
7. On the Default Sync Class Operations menu, press **Enter** to accept the default. The Default Sync Class determines how all entries that do not match any other Sync Class are handled.
8. Review the configuration, and press **Enter** to write the configuration to the server.

Use the `dsconfig` tool to make changes to this configuration. See Configuring the PingDataSync Server for configuration options and details.

Considerations for synchronizing from a database source

When synchronizing from a database to a directory or RDBMS server, the following are recommended:

- **Identify Database Entry Types** – Identify the database entry types that will be synchronized, and:
    - Set the `database-entry-type` property on the JDBC Sync Source (this is required), and make sure the entry types are what the triggers are inserting into the change tracking mechanism.
    - Create a Sync Class per entry type, and set different mappings and rules for each one.
- For each Sync Class, do the following:
    - Make sure the `include-filter` property is set to match the entry type.
    - Create a specific attribute mapping for every database column to be synchronized to an LDAP attribute and set it on the Sync Class. If this is done, the script will not have to know about the schema on the directory side.
    - Create a DN map that recognizes the DNs generated by the script and maps them to the correct location at the destination.
    - Remove the default value of "-all-" from the `auto-mapped-source-attribute` property on the Sync Class, and replace it with the value `objectClass`. The object class for the fetched source entry is determined by the scripted layer. Values from the database should not be automatically mapped to an attribute with the same name, except the `objectclass` attribute, which should map directly for CREATE operations. If this is not done, an error is generated.
    - Change the `destination-correlation-attributes` property to contain the attributes that uniquely represent the database entries on the directory server destination.
- **Avoid Bidirectional Loopback** – Set the `ignore-changes-by-[user|dn]` property on both Sync Sources when configuring for bidirectional synchronization, to make sure that changes are not looped back by the PingDataSync Server.

See Use the create-sync-pipe tool to configure synchronization for details about creating the Sync Pipe.

Synchronize a specific list of database elements

The `resync` command enables synchronizing a specific set of database keys that are read from a JDBC Sync Source file using the `--sourceInputFile` option. The contents of the file are passed line-by-line into the `listAllEntries()` method of the `JDBCSyncSource` extension, which is used for the Sync Pipe. The method processes the input and returns `DatabaseChangeRecord` instances based on the input from the file.

Perform the following steps to synchronize a specific list of database elements using the `resync` tool:

1. Create a file of JDBC Sync Source elements. There is no set format for the file, but it typically contains a list of primary keys or SQL queries. For example, create a file containing a list of primary keys and save it as `sourceSQL.txt`.

   ```
   user.0
   user.1
   user.2
   user.3
   ```

2. Run the `resync` command with the `--sourceInputFile` option to run on individual primary keys in the file.

   ```
   $ bin/resync --pipe-name "dbsync-pipe" \
     --sourceInputFile sourceSQL.txt
   ```

3. If searching for a specific type of database entry, use the `--entryType` option that matches one of the configured entry types in the JDBCSyncSource.

   ```
   $ bin/resync --pipe-name "dbsync-pipe" \
     --entryType account \
     --sourceInputFile sourceSQL.txt
   ```

Chapter 6: Synchronize through PingDirectory Proxy Servers

Because most data centers deploy directory servers in a proxied environment, the PingDataSync Server can also synchronize data through a proxy server in both load-balanced and entry-balancing deployments. The following proxy endpoints are supported:

- Ping Identity PingDirectory Proxy Servers
- Alcatel-Lucent 8661 Directory Proxy Servers

This chapter details a Sync-through-Proxy deployment and provides background information on how it works. Before setting up the PingDataSync Server, review the *Ping Identity PingDirectory Proxy Server Administration Guide* for information about the PingDirectory Proxy Server.

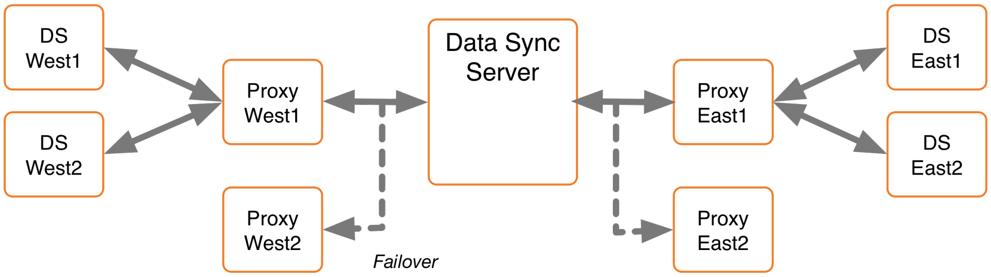
Topics include:

Synchronization through a Proxy Server overview

To handle data synchronization through a proxy server, PingData servers have a `cn=changelog` state management system that supports a token-based API. In a standard, non-proxied configuration, the PingDataSync Server polls the source server for changes, determines if a change is necessary, and fetches the full entry from the source. Then, it finds the corresponding entry in the destination endpoint using correlation rules and applies the minimal set of changes. The server fetches and compares the full entries to make sure it does not synchronize any stale data from the change log.

In a proxied environment, the PingDataSync Server passes the request through a proxy server to the backend set of directory servers. The PingDataSync Server uses the highest priority proxy server designated in its endpoint server configuration and can use others in the event of a failover.

The following illustrates a deployment with two endpoints consisting of a proxy server deployment in front of the backend set of directory servers.



Synchronization Through Proxy Example

Change log operations

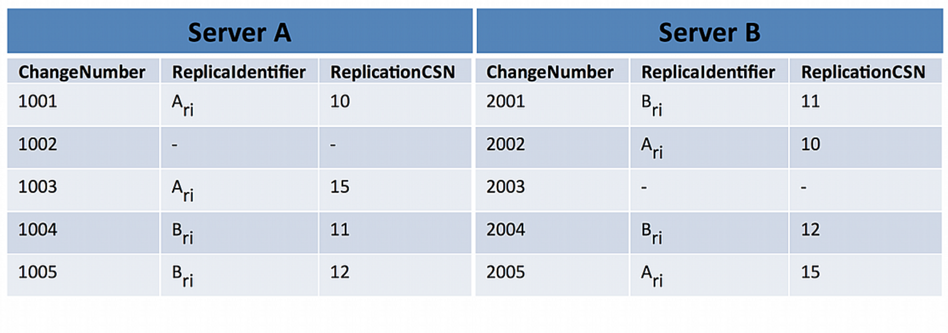
When the PingDataSync Server runs a poll for any changes, it sends a get change log batch extended request to the `cn=changelog` backend. The batch request looks for entries in the change log and asks for the server ID, change number, and replica state for each change. The PingDirectory Proxy Server routes the request to a directory server instance, which then returns a changed entry plus a token identifying the server ID, change number and replica state for each change. The PingDirectory Proxy Server then sends a get change log batch response back to the PingDataSync Server with this information. For entry-balancing deployments, the PingDirectory Proxy Server must "re-package" the directory server tokens into its own proxy token to identify the specific data set.

The first time that the PingDataSync Server issues the batch request, it also issues a get server ID request to identify the specific server ID that is processing the extended request. The PingDirectory Proxy Server routes the request to the directory server instance, and then returns a server ID in the response. With the next request, the PingDataSync Server sends a 'route to server' request that specifies the server instance to access again in this batch session. It also issues a server ID request in the event that the particular server is down. This method avoids round-robin server selection and provides more efficient overall change processing.

Directory Server and PingDirectory Proxy Server tokens

The Directory Server maintains a change log database index to determine when to resume sending changes (for ADD, MODIFY, or DELETE operations) in its change log. While a simple stand-alone directory server can track its resume point by the last change number sent, replicated servers or servers deployed in entry balancing environments have a different change number ordering in its change log because updates can come from a variety of sources.

The following illustrates two change logs in two replicated directory servers, server A and B. "A" represents the replica identifier for a replicated subtree in Server A, and "B" represents the replica identifier for the same replicated subtree in server B. The replica identifiers with a hyphen ("-") mark any local, non-replicated but different changes. While the two replicas record all of the changes, the two change logs have two different change number orderings because updates come in at different times.



Server A			Server B		
ChangeNumber	ReplicaIdentifier	ReplicationCSN	ChangeNumber	ReplicaIdentifier	ReplicationCSN
1001	A_{ri}	10	2001	B_{ri}	11
1002	-	-	2002	A_{ri}	10
1003	A_{ri}	15	2003	-	-
1004	B_{ri}	11	2004	B_{ri}	12
1005	B_{ri}	12	2005	A_{ri}	15

Different Change Number Order in Two Replicated Change Logs

To track the change log resume position, the Directory Server uses a change log database index to identify the latest change number position corresponding to the highest replication CSN number for a given replica. This information is encapsulated in a directory server token and returned in the get change log batch response to the PingDirectory Proxy Server. The token has the following format:

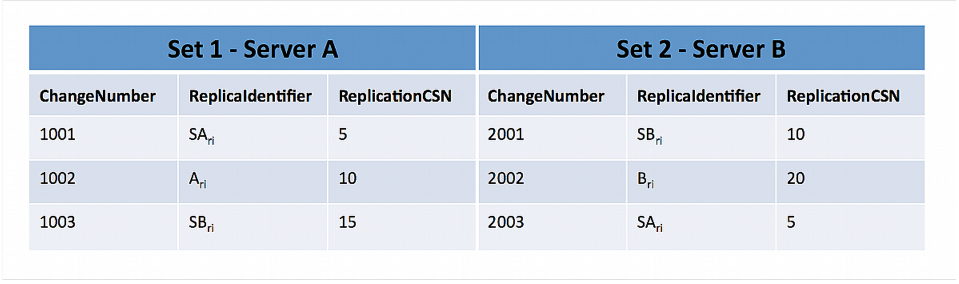
```
Directory Server Token: server ID, changeNumber, replicaState
```

For example, if the PingDirectory Proxy Server sends a request for any changed entries, and the directory servers return the change number 1003 from server A and change number 2005 from server B, then each directory server token would contain the following information:

```
Directory Server Token A:
  serverID A, changeNumber 1003, replicaState {15(A)}

Directory Server Token B:
  serverID B, changeNumber 2005, replicaState {12(B), 15(A)}
```

Change log tracking in entry balancing deployments

Change log tracking can become more complex in that a shared area of data can exist above the entry-balancing base DN in addition to each backend set having its own set of changes and tokens. In the following figure, Server A belongs to an entry-balancing set 1, and server B belonging to an entry-balancing set 2. Shared areas that exist above the entry-balancing base DN are assumed to be replicated to all servers. "SA" represents the replica identifier for that shared area on Server A and "SB" represents the replica identifier for the same area on Server B.



Set 1 - Server A			Set 2 - Server B		
ChangeNumber	ReplicaIdentifier	ReplicationCSN	ChangeNumber	ReplicaIdentifier	ReplicationCSN
1001	SA_{ri}	5	2001	SB_{ri}	10
1002	A_{ri}	10	2002	B_{ri}	20
1003	SB_{ri}	15	2003	SA_{ri}	5

Different Change Number Order in Two Replicated Change Logs

The PingDirectory Proxy Server cannot pass a directory server token from the client to the directory server and back again. In an entry-balancing deployment, the PingDirectory Proxy Server must maintain its own token mechanism that associates a directory server token (`changeNumber`, `replicaIdentifier`, `replicaState`) to a particular backend set.

```
Proxy Token:
backendSetID 1: ds-token 1 (changeNumber, replicaIdentifier, replicaState)
backendSetID 2: ds-token 2 (changeNumber, replicaIdentifier, replicaState)
```

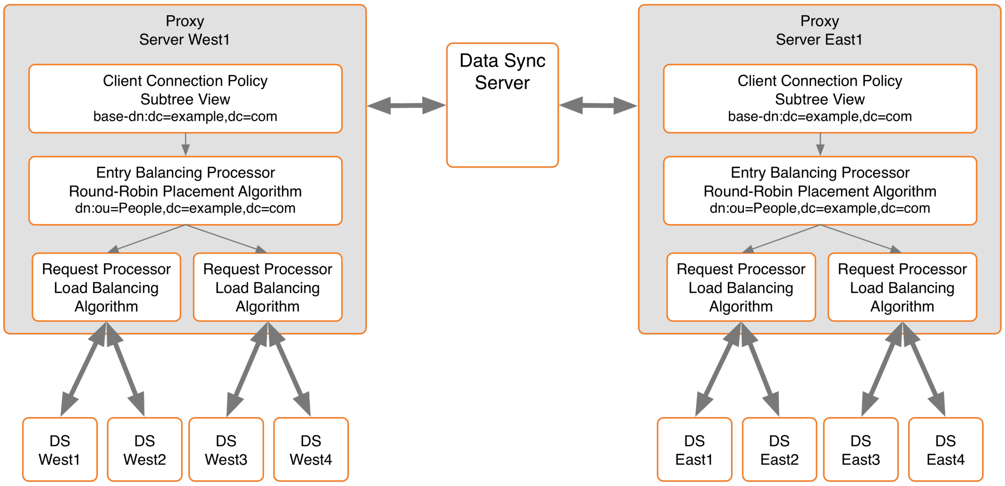
For example, if the PingDirectory Proxy Server returned change 1002 from Server A and change 2002 from Server B, then the Proxy token would contain the following:

```
Proxy Token:
backendSetID 1: ds-token-1 {serverID A, changeNumber 1002, replicaState (5
(SA), 15(A)}
backendSetID 2: ds-token-2 {serverID B, changeNumber 2002, replicaState (10
(SB), 20(B)}
```

For each change entry returned by a backend, the PingDirectory Proxy Server must also decide whether it is a duplicate of a change made to the backend set above the entry-balancing base. If the change is a duplicate, then it is discarded. Otherwise, any new change is returned with a new value of the proxy token.

Example configuration

This following configures synchronization through a proxy and use two endpoints consisting of a PingDirectory Proxy Server with a backend set of Directory Servers: both sets are replicated. The PingDirectory Proxy Server uses an entry-balancing environment for the DN `ou=People,dc=example,dc=com` and provides a subtree view for `dc=example,dc=com` in its client connection policy. For this example, communication is over standard LDAP and failover servers are not installed or designated in the PingDataSync Server.


Proxy Server West1
Client Connection Policy Subtree View base-dn:dc=example,dc=com
Entry Balancing Processor Round-Robin Placement Algorithm dn:ou=People,dc=example,dc=com
Request Processor Load Balancing Algorithm
Request Processor Load Balancing Algorithm
Data Sync Server
Proxy Server East1
Client Connection Policy Subtree View base-dn:dc=example,dc=com
Entry Balancing Processor Round-Robin Placement Algorithm dn:ou=People,dc=example,dc=com
Request Processor Load Balancing Algorithm
Request Processor Load Balancing Algorithm
DS West1
DS West2
DS West3
DS West4
DS East1
DS East2
DS East3
DS East4

Example Synchronization Through Proxy Configuration

Configure the source Directory Server

The following procedures configure a backend set of directory servers. The procedure is the same for the source servers and the destination servers in a synchronization topology. For directory server installation and configuration details, see the *Ping Identity Directory Server Administration Guide*.

1. On each backend Directory Server that will participate in synchronization, enable the change log database, either from the command line or by using a `dsconfig` batch file.

```
$ dsconfig --no-prompt set-backend-prop \
  --backend-name changelog \
  --set enabled:true
```

2. Stop the server if it is running, and import the dataset for the first backend set into the first server in the backend set prior to the import.

```
$ bin/stop-server
```

```
$ bin/import-ldif --backendID userRoot --ldifFile ../dataset.ldif
```

```
$ bin/start-server
```

3. On the first server instance in the first backend set, configure replication between this server and the second server in the same backend set.

```
$ bin/dsreplication enable --host1 ldap-west-01.example.com \
  --port1 389 \
  --bindDN1 "cn=Directory Manager" \
  --bindPassword1 password \
  --replicationPort1 8989 \
  --host2 ldap-west-02.example.com \
  --port2 389 \
  --bindDN2 "cn=Directory Manager" \
  --bindPassword2 password \
  --replicationPort2 9989 \
  --adminUID admin \
  --adminPassword admin \
  --baseDN dc=example,dc=com \
  --no-prompt
```

4. Initialize the second server in the backend set with data from the first server in the backend set. This command can be run from either instance.

```
$ bin/dsreplication initialize \
  --hostSource ldap-west-01.example.com \
  --portSource 389 \
  --hostDestination ldap-west-02.example.com \
  --portDestination 389 \
  --baseDN "dc=example,dc=com" \
  --adminUID admin \
  --adminPassword admin \
  --no-prompt
```

5. Run the following command to check replica status.

```
$ bin/dsreplication status \
  --hostname ldap-west-01.example.com \
  --port 389 \
  --adminPassword admin \
  --no-prompt
```

6. Repeat steps 3 through 6 (import, enable replication, initialize replication, check status) for the second backend set.

Configure a Proxy Server

The following procedures configure a proxy server, including defining the external servers and configuring the client-connection policy. The procedure is the same for the source servers and the destination servers in a synchronization topology. For additional changes, use the `dsconfig` tool. For proxy installation and configuration details, see the *Ping Identity PingDirectory Proxy Server Administration Guide*.

1. From the PingDirectory Proxy Server root directory, run the `prepare-external-server` command to set up the `cn=Proxy User` account for access to the backend directory servers. The server tests the connection and creates the `cn=Proxy User` account.

   ```
   $ bin/prepare-external-server --no-prompt \
     --hostname ldap-west-01.example.com \
     --port 389 --bindDN "cn=Directory Manager" \
     --bindPassword password \
     --proxyBindDN "cn=Proxy User,cn=Root DNs,cn=config" \
     --proxyBindPassword pass \
     --baseDN "dc=example,dc=com"
   ```

2. Repeat step 1 for any other directory server instances.

3. Run the `dsconfig` command to define the external servers and their types. For this example, round-robin load balancing algorithms are defined, which do not require health checks or locations to be specified.

   ```
   $ bin/dsconfig --no-prompt create-external-server \
     --server-name ldap-west-01 \
     --type "ping-identity-ds" \
     --set "server-host-name:ldap-west-01.example.com" \
     --set "server-port:389" \
     --set "bind-dn:cn=Proxy User" \
     --set "password:password" \
     --bindDN "cn=Directory Manager" \
     --bindPassword pxy-pwd
   ```

   ```
   $ bin/dsconfig --no-prompt create-external-server \
     --server-name ldap-west-02 \
     --type "ping-identity-ds" \
     --set "server-host-name:ldap-west-02.example.com" \
     --set "server-port:389" \
     --set "bind-dn:cn=Proxy User" \
     --set "password:password" \
     --bindDN "cn=Directory Manager" \
     --bindPassword pxy-pwd
   ```

   ```
   $ bin/dsconfig --no-prompt create-external-server \
     --server-name ldap-west-03 \
     --type "ping-identity-ds" \
     --set "server-host-name:ldap-west-03.example.com" \
     --set "server-port:389" \
     --set "bind-dn:cn=Proxy User" \
     --set "password:password" \
     --bindDN "cn=Directory Manager" \
     --bindPassword pxy-pwd
   ```

   ```
   $ bin/dsconfig --no-prompt create-external-server
     --server-name ldap-west-04 \
     --type "ping-identity-ds" \
     --set "server-host-name:ldap-west-04.example.com" \
     --set "server-port:389" \
   ```

```
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

4. Create a load-balancing algorithm for each backend set.

```
$ bin/dsconfig --no-prompt create-load-balancing-algorithm \
  --algorithm-name "test-lba-1" \
  --type "round-robin" --set "enabled:true" \
  --set "backend-server:ldap-west-01" \
  --set "backend-server:ldap-west-02" \
  --set "use-location:false" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-load-balancing-algorithm \
  --algorithm-name "test-lba-2" \
  --type "round-robin" --set "enabled:true" \
  --set "backend-server:ldap-west-03"
  --set "backend-server:ldap-west-04"
  --set "use-location:false" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

5. Configure the proxying request processors, one for each load-balanced directory server set. A request processor provides the logic to either process the operation directly, forward the request to another server, or hand off the request to another request processor.

```
$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "proxying-processor-1" --type "proxying" \
  --set "load-balancing-algorithm:test-lba-1" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "proxying-processor-2" --type "proxying" \
  --set "load-balancing-algorithm:test-lba-2" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

6. Define an entry-balancing request processor. This request processor is used to distribute entries under a common parent entry among multiple backend sets. A backend set is a collection of replicated directory servers that contain identical portions of the data. Multiple proxying request processors are used to process operations.

   Next, define the placement algorithm, which selects the server set to use for new add operations to create new entries. In this example, a round-robin placement algorithm forwards LDAP add requests to backends sets.

```
$ bin/dsconfig --no-prompt create-placement-algorithm \
  --processor-name "entry-balancing-processor" \
  --algorithm-name "round-robin-placement" \
  --set "enabled:true" \
  --type "round-robin" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

7. Define the subtree view that specifies the base DN for the entire deployment.

```
$ bin/dsconfig --no-prompt create-subtree-view \
  --view-name "test-view" \
  --set "base-dn:dc=example,dc=com" \
  --set "request-processor: entry-balancing-processor" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

8. Finally, define a client connection policy that specifies how the client connects to the proxy server.

```
$ bin/dsconfig --no-prompt set-client-connection-policy-prop \
  --policy-name "default" \
  --add "subtree-view:test-view" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

Configuring the PingDataSync Server

Configure the PingDataSync Server once the PingDirectory Proxy Server and its backend set of Directory Server instances are configured and fully functional for each endpoint, which are labeled as `ldap-west` and `ldap-east` in this example. For information on installing and configuring the PingDataSync Server, see Installing the PingDataSync Server.

1. From the PingDataSync Server root directory, run the `create-sync-pipe-config` tool.

```
$ bin/create-sync-pipe-config
```

2. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.
3. On the Synchronization Mode menu, press **Enter** to select Standard mode.
4. On the Synchronization Directory menu, choose the option for one-way or bidirectional synchronization.
5. On the First Endpoint Type menu, enter the number for the type of backend data store for the first endpoint. In this example, type the number corresponding to the PingDirectory Proxy Server.

```
>>>> First Endpoint Type
Enter the type of data store for the first endpoint:
1) Ping Identity Directory Server
2) Ping Identity Directory Proxy Server
3) Alcatel-Lucent Directory Server
4) Alcatel-Lucent Proxy Server
5) Sun Directory Server
```

```
6) Microsoft Active Directory
7) Microsoft SQL Server
8) Oracle Database
9) Custom JDBC

b) back
q) quit

Enter choice [1]: 2
```

6. Enter a descriptive name for the first endpoint.
7. Enter the base DN where the PingDataSync Server can search for the entries on the first endpoint server.
8. Specify the type of security when communicating with the endpoint server.
9. Enter the hostname and port of the endpoint server. The PingDataSync Server tests the connection. Repeat this step if configuring another server for failover.
10. Enter the Sync User account that will be used to access the endpoint server, or press **Enter** to accept the default `cn=Sync User,cn=Root DNs,cn=config`. Enter a password for the account.
11. The first endpoint deployment is defined using the PingDirectory Proxy Server (`ldap-west`). Repeat steps 5-10 to define the second proxy deployment (`ldap-east`) on the PingDataSync Server.
12. Prepare the endpoint servers in the topology. This step confirms that the Sync User account is present on each server and can communicate between the PingDataSync Server and the PingDirectory Proxy Servers. In addition to preparing the PingDirectory Proxy Server, the PingDataSync Server prepares the backend set of directory servers as the proxy server passes through the authorization to access these servers.
13. Repeat the previous step to prepare the second endpoint server. If other servers have not been prepared, make sure that they are prior to synchronization.
14. Define the Sync Pipe from proxy 1 to proxy 2. In this example, accept the default "Ping Identity Proxy 1 to Ping Identity Proxy 2."
15. To customize on a per-entry basis how attributes get synchronized, define one or more Sync Classes. Create a Sync Class for the special cases, and use the default Sync Class for all other mappings.
16. For the default Sync Class Operations, specify the operations that will be synchronized.
17. Review the configuration settings, and write the configuration to the PingDataSync Server in the `sync-pipe-cfg.txt` file.

Test the configuration

If the `create-sync-pipe-config` tool was not used to create the synchronization configuration, two properties must be verified on each endpoint: `proxy-server` and `use-changelog-batch-request`. The `proxy-server` property should specify the name of the proxy server. The `use-changelog-batch-request` should be set to `true` on the Sync Source only. The `use-changelog-batch-request` is not available on the destination endpoint.

The PingDataSync Server connection parameters (hostname, port, bind DN, and bind password) are required.

1. The following commands check the properties on a Sync Source.

   On the Sync Source:

   ```
   $ bin/dsconfig --no-prompt \
     get-sync-source-prop \
     --source-name "Ping Identity Proxy 1" \
     --property "proxy-server" \
     --property "use-changelog-batch-request"
   ```

   On the Sync Destination:

   ```
   $ bin/dsconfig --no-prompt \
     get-sync-source-prop \
     --source-name "Ping Identity Proxy 2" \
     --property "proxy-server"
   ```

2. From the server root directory, run the `dsconfig` command to set a flag indicating that the endpoints are PingDirectory Proxy Servers:

   ```
   $ bin/dsconfig --no-prompt \
     set-sync-source-prop \
     --source-name "Ping Identity Proxy 1" \
     --set proxy-server:ldap-west-01 \
     --set use-changelog-batch-request:true
   ```

   ```
   $ bin/dsconfig --no-prompt \
     set-sync-source-prop \
     --source-name "Ping Identity Proxy 2" \
     --set proxy-server:ldap-east-01
   ```

3. Run the `resync --dry-run` command to test the configuration settings for each Sync Pipe and debug any issues.

   ```
   $ bin/resync --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
   --dry-run
   ```

4. Run `realtime-sync set-startpoint` to initialize the starting point for synchronization.

   ```
   $ realtime-sync set-startpoint --end-of-changelog \
     --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2" \
     --port 389 \
     --bindDN "cn=Directory Manager" \
     --bindPassword password
   ```

Note
For synchronization through proxy deployments, the `--change-number` option cannot be used with the `realtime-sync set-startpoint` command, because the PingDataSync Server cannot retrieve specific change numbers from the backend directory servers. Use the `--change-sequence-number`, `--end-of-changelog`, or other options available for the tool.

5. Run the `resync` command to populate data on the endpoint destination server if necessary.

```
$ bin/resync --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
\
  --numPasses 3
```

6. Start the Sync Pipe using the `realtime-sync start` command.

```
$ bin/realtime-sync start \
  --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
```

7. Monitor the PingDataSync Server using the status commands and logs.

Index the LDAP changelog

The PingData Directory Server and the Alcatel-Lucent 8661 Directory Server (3.0 or later) both support attribute indexing in the changelog backend to enable get changelog batch requests to filter results that include only changes of specific attributes. For example, in an entry balanced proxy deployment, the PingDataSync Server sends a get changelog batch request to the Proxy Server, which will send out individual requests to each backend server.

Each directory server that receives a request must iterate over the whole range of changelog entries, and then match entries based on search criteria for inclusion in the batch. The majority of this processing involves determining whether a changelog entry includes changes to a particular attribute or set of attributes, or not. Changelog indexing can dramatically speed up throughput when targeting specific attributes.

Attribute indexing is configured using the `index-include-attribute` and `index-exclude-attribute` properties on the changelog backend. The properties can accept the specific attribute name or special LDAP values "*" to specify all user attributes or "+" to specify all operational attributes.

Perform the following steps to configure changelog indexing:

1. On all source directory servers, enable changelog indexing for the attributes that will be synchronized. Use the `index-include-attribute` and `index-exclude-attribute` properties. The following example specifies that all user attributes (`index-include-attribute:*`) be indexed in the changelog, except the description and location attributes (`index-exclude-attribute:description` and `index-exclude-attribute:location`).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "index-include-attribute:*" \
  --set "index-exclude-attribute:description \
  --set "index-exclude-attribute:location
```

Note
There is little performance and disk consumption penalty when using index-include-attribute:* with a combination of index-exclude-attribute properties, instead of explicitly defining each attribute using index-include-attribute. The only cautionary note about using index-include-attribute:* is to be careful that unnecessary attributes get indexed.

2. On the PingDataSync Server, configure the auto-map-source-attributes property to specify the mappings for the attributes that need to be synchronized.

The PingDataSync Server will write a NOTICE message to the error log when the Sync Pipe first starts, indicating whether the server is using changelog indexing or not.

```
[30/Mar/2016:13:21:36.781 -0500] category=SYNC severity=NOTICE
msgID=1894187256 msg="Sync Pipe 'TestPipe' is not using changelog indexing on
the source server"
```

Changelog synchronization considerations

If the Sync Source is configured with use-changelog-batch-request=true, the PingDataSync Server will use the get changelog batch request to retrieve changes from the LDAP changelog. This extended request can contain an optional set of selection criteria, which specifies changelog entries for a specific set of attributes.

The PingDataSync Server takes the union of the source attributes from DN mappings, attribute mappings, and the auto-mapped-source-attributes property on the Sync Class to create the selection criteria. However, if it encounters the value "-all-" in the auto-mapped-source-attributes property, it cannot make use of selection criteria because the Sync Pipe is interested in all possible source attributes.

When the Directory Server receives a get changelog request that contains selection criteria, it returns changelog entries for one or more of the attributes that meet the criteria.

- For ADD and MODIFY changelog entries, the changes must include at least one attribute from the selection criteria.
- For MODDN changelog entries, one of the RDN attributes must match the selection criteria.
- For DELETE changelog entries, one of the deletedEntryAttrs much match the selection criteria.

If auto-mapped is not set to all source attributes, at least one should be configured to show up in the deletedEntryAttrs (with the changelog-deleted-entry-include-attribute property on the changelog backend).

Another way to do this is to set use-reversible-form to true on the changelog backend. This includes all attributes in the deletedEntryAttrs.

Chapter 7: Synchronize in Notification Mode

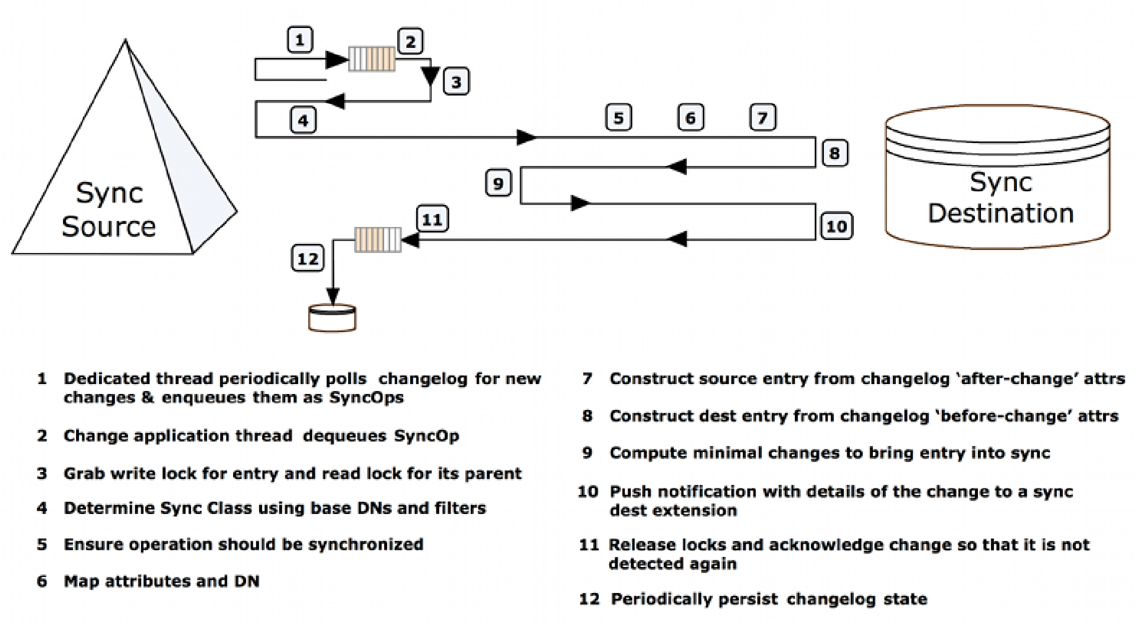
The PingDataSync Server supports a notification synchronization mode that transmits change notifications on a source endpoint to third-party destination applications. As with standard mode, notifications can be filtered based on the type of entry that was changed, the specific attributes that were changed, and the type of change (ADD, MODIFY, DELETE). The PingDataSync Server can send a notification to arbitrary endpoints by using a custom server extension.

Topics include:

Notification mode overview

The PingDataSync Server supports standard and notification synchronization modes. Notification Mode polls the directory server's LDAP change log for changes on any entry but skips the fetch and compare phases of processing of Standard Mode. Instead, the Sync Destination is notified of the change regardless of the current state of that entry at the source or destination. The PingDataSync Server accesses state information on the change log to reconstruct the before-and-after values of any modified attribute (for example, for MODIFY change operation types). It passes in the change information to a custom server extension based on the Server SDK.

Third-party libraries can be employed to customize the notification message to an output format required by the client application or service. For example, the server extension can use a third-party XML parsing library to convert the change notifications to a SOAP XML format. Notification mode can only be used with an Directory Server, Alcatel-Lucent 8661 Directory Server, PingDirectory Proxy Server, or Alcatel-Lucent 8661 Directory Proxy Server as the source endpoint.



1 Dedicated thread periodically polls changelog for new changes & enqueues them as SyncOps

2 Change application thread dequeues SyncOp

3 Grab write lock for entry and read lock for its parent

4 Determine Sync Class using base DNs and filters

5 Ensure operation should be synchronized

6 Map attributes and DN

7 Construct source entry from changelog 'after-change' attrs

8 Construct dest entry from changelog 'before-change' attrs

9 Compute minimal changes to bring entry into sync

10 Push notification with details of the change to a sync dest extension

11 Release locks and acknowledge change so that it is not detected again

12 Periodically persist changelog state

Notification Mode Synchronization Change Flow

The PingDataSync Server can use notification mode with any type of endpoint; therefore, it is not an absolute requirement to have a custom server extension in your system. For example, it is possible to set up a notification Sync Pipe between two LDAP server endpoints.

Implementation Considerations

Before implementing and configuring a Sync Pipe in notification mode, answer the following questions:

- What is the interface to client applications?
- What type of connection logic is required?
- How will the custom server extension handle timeouts and connection failures?
- What are the failover scenarios?
- What data needs to be included in the change log?
- How long do the change log entries need to be available?
- What are the scalability requirements for the system?
- What attributes should be used for correlation?
- What should happen with each type of change?
- What mappings must be implemented?

Use the Server SDK and LDAP SDK

To support notification mode, the Server SDK provides a `SyncDestination` extension to synchronize with any client application. The PingDataSync Server engine processes the notification and makes it available to the extension, which can be written in Java or Groovy. This generic extension type can also be used for standard synchronization mode.

Similar to database synchronization, the custom server extension is stored in the `<server-root>/lib/groovy-scripted-extensions` folder (for Groovy-based extensions) or the jar file in the `<server-root>/lib/extensions` folder (for Java-based extensions) prior to configuring the PingDataSync Server for notification mode. Groovy scripts are compiled and loaded at runtime.

The Server SDK's `SyncOperation` interface represents a single synchronized change from the Sync Source to the Sync Destination. The same `SyncOperation` object exists from the time a change is detected, through when the change is applied at the destination.

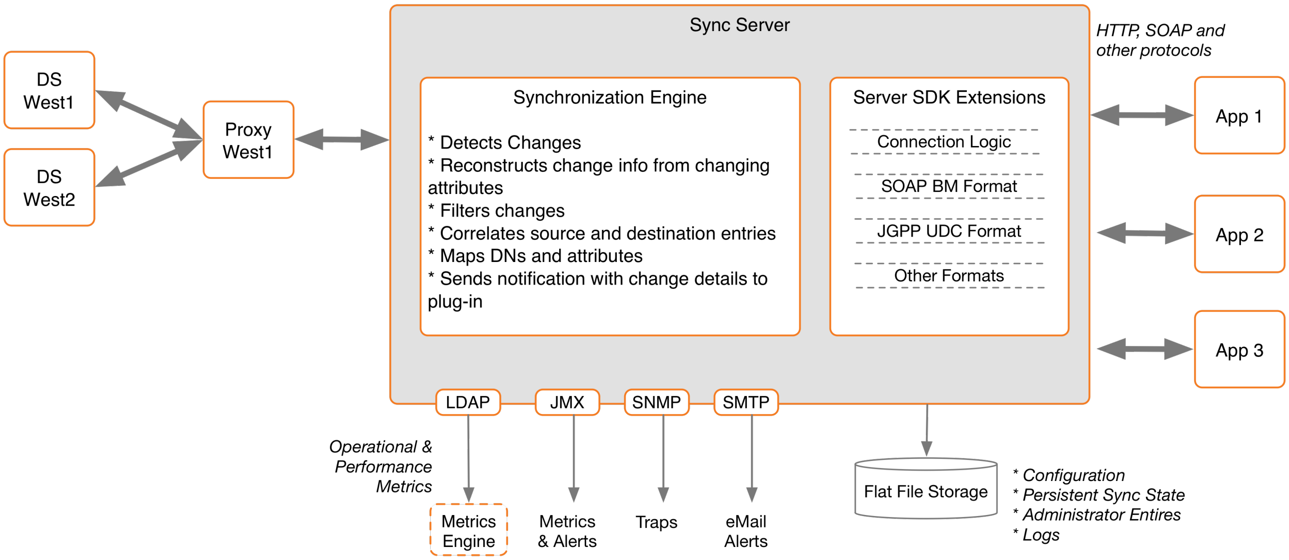
The LDAP SDK's `UnboundIDChangelogEntry` class (in the `com.unboundid.ldap.sdk.unboundidds` package) has high level methods to work with the `ds-changelog-before-value`, `ds-changelogafter-values`, and `ds-changelog-entry-key-attr-values` attributes. The class is part of the commercial edition of the LDAP SDK for Java and is installed automatically with the PingDataSync Server. For detailed information and examples, see the LDAP SDK Javadoc.

Notification mode architecture

Notification mode, a configuration setting on the Sync Pipe, requires a one-way directional Sync Pipe from a source endpoint topology to a target client application. The PingDataSync Server detects the changes in the Directory Server’s LDAP change log, filters the results specified in the Sync Classes, applies any DN and attribute mappings, then reconstructs the change information from the change log attributes. A server extension picks up the notification arguments from the `SyncOperation` interface (part of the Server SDK) and converts the data to the desired output format. The server extension establishes the connections and protocol logic to push the notification information to the client applications or services. All of the operations, administration, and management functions available in standard mode, such as monitoring, (LDAP, JMX, SNMP), alerts (JMX, SNMP, SMTP), and logging features are the same for notification mode.

> **Note**
> The Server SDK includes documentation and examples on how to create a directory server extension to support notification mode.

For a given entry, the PingDataSync Server sends notifications in the order that the changes occurred in the change log even if a modified attribute has been overwritten by a later change. For example, if an entry’s `telephoneNumber` attribute is changed three times, three notifications will be sent in the order they appeared in the change log.



Notification Mode Architecture

Sync Source requirements

A separate Sync Pipe is required for each client application that should receive a notification. The Sync Sources must consist of one or more instances of the following directory or proxy servers with the PingDataSync Server:

- Ping IdentityDirectory Server and PingDirectory Proxy Server (version 3.0.5 or later)
- Alcatel-Lucent 8661 Directory Server
- The Sync Destination can be of any type

> **Note** While the PingDirectory Proxy Server and Alcatel-Lucent 8661 Directory Proxy Server can front other vendor’s directory servers, such as Active Directory and Sun DSEE, for processing LDAP operations, the PingDataSync Server cannot synchronize changes from these sources through a proxy

server. Synchronizing changes directly from Active Directory and Sun DSEE cannot be done with notification mode.

Failover Capabilities

For sync source failovers, configure replication between the Directory Servers to ensure data consistency between the servers. A PingDirectory Proxy Server can also front the backend Directory Server set to redirect traffic, if connection to the primary server fails. A PingDirectory Proxy Server must be used for synchronizing changes in an entry-balancing environment. Once the primary Directory Server is online, it assumes control with no information loss as its state information is kept across the backend Directory Servers.

For sync destination failovers, connection retry logic must be implemented in the server extension, which will use the Sync Pipe’s advanced property settings to retry failed operations. There is a difference between a connection retry and an operation retry. An extension should not retry operations because the PingDataSync Server does this automatically. However, the server extension is responsible for re-establishing connections to a destination that has gone down, or failing over to an alternate server. The server extension can also be designed to trigger its own error- handling code during a failed operation.

For PingDataSync Server failovers, the secondary PingDataSync Servers will be at or slightly behind the state where the primary server initiated a failover. Both primary and secondary PingDataSync Servers track the last failed acknowledgement, so once the primary server fails over to a secondary server, the secondary server will not miss a change.

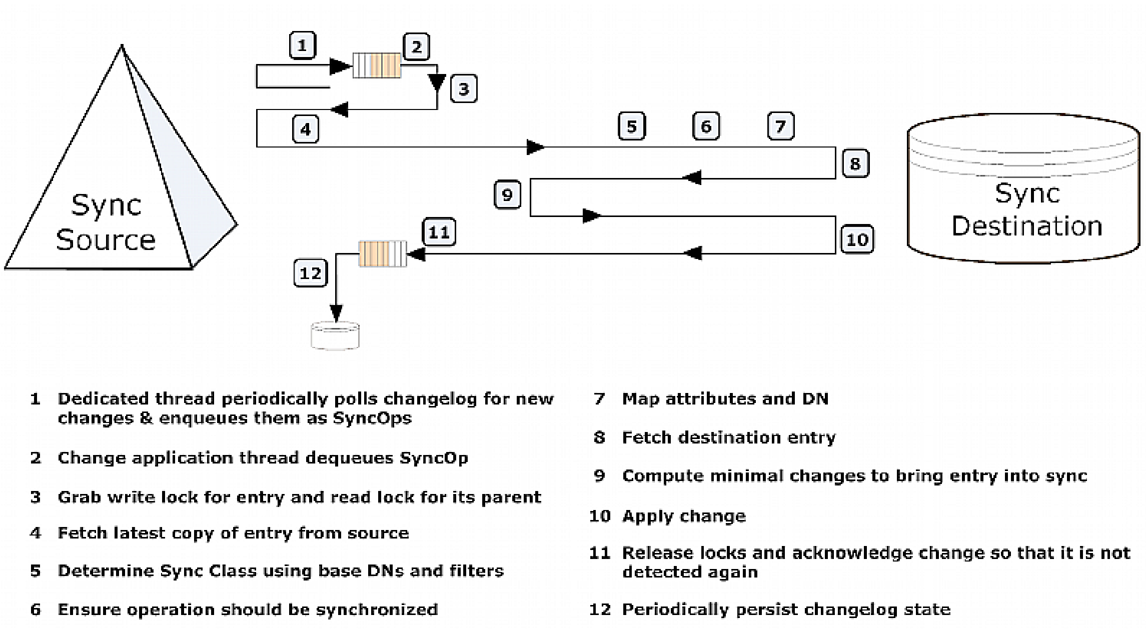
Note
If failover is a concern between PingDataSync Servers, change the `sync-failover-polling-interval` property from 7500 ms to a smaller value. This will result in a quicker failover, but will marginly increase traffic between the two PingDataSync Servers. The `sync-failover-connection-timeout` and `sync-failover-response-timeout` properties may also be updated to use different failover timeout durations. Use `dsconfig` to access the property on the Global Sync Configuration menu.

Notification Sync Pipe change flow

Multi-threaded Sync Pipes allow the PingDataSync Server to process multiple notifications in parallel in the same manner as synchronizing changes in standard mode, which increases throughput and offsets network latency. A single change-detection thread pulls in batches of change log entries and queues them internally. To guarantee consistency, the PingDataSync Server’s internal locking mechanisms ensure the following properties:

- Changes to the same entry will be processed in the same order that they appear in the change log.
- Changes to parent entries will be processed before changes to its children.
- Changes to entries with the same RDN value are handled sequentially.

The number of concurrent threads is configurable on the Sync Pipe using the `num-worker-threads` property. In general, single-threading should be avoided.



1 Dedicated thread periodically polls changelog for new changes & enqueues them as SyncOps

2 Change application thread dequeues SyncOp

3 Grab write lock for entry and read lock for its parent

4 Fetch latest copy of entry from source

5 Determine Sync Class using base DNs and filters

6 Ensure operation should be synchronized

7 Map attributes and DN

8 Fetch destination entry

9 Compute minimal changes to bring entry into sync

10 Apply change

11 Release locks and acknowledge change so that it is not detected again

12 Periodically persist changelog state

Notification Sync Pipe Change Flow

Configure Notification mode

The PingDataSync Server supports notification mode with the following components:

Use the create-sync-pipe-config tool

The `create-sync-pipe-config` tool supports the configuration of notification mode. Any pre-existing Sync Sources can be read from the local configuration (in the `config.ldif` file).

No resync command functionality

The `resync` function is disabled on a Sync Pipe in notification mode as its functionality is not supported in this implementation. Notification mode views the directory server's change log as a rolling set of data that pushes out change notifications to its target application.

LDAP change log features required for notifications

The Directory Server and the Alcatel-Lucent 8661 Directory Server require the following advanced global change log properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

These properties are enabled and configured during the `create-sync-pipe-config` configuration process on the PingDataSync Server. The properties can also be enabled on the directory servers using the `dsconfig` advanced properties setting on the Backend Changelog menu.

changelog-include-key-attribute

The changelog-include-key-attribute property specifies one or more attributes that should always be included in the change log entry. These are attributes needed to correlate entries between the source and destination, such as uid, employeeNumber, or mail. These properties are also needed for evaluating any filters in the Sync Class. For example, if notifications are only sent for user entries, and the Sync Class included the filter (objectclass=people), the objectclass attribute must be configured as a changelog-include-key-attribute so that the Sync Pipe can evaluate the inclusion criteria when processing the change. In standard mode, values needed in the filter are read from the entry itself after it is fetched instead of from the changelog entry. These attributes are always included in a change log entry, also called a change record, regardless if they have changed or not.

The changelog-include-key-attribute property causes the current (after-change) value of the specified attributes to be recorded in the ds-changelog-entry-key-attr-values attribute on the change log entry. This applies for all change types. During a delete operation, the values are from the entry before it was deleted. The key values are recorded on every change and override any settings configured in the changelog-include-attribute, changelog-exclude-attribute, changelog-deleted-entry-include-attribute, or changelog-deleted-entry-exclude-attribute properties in the directory server changelog (see the *Ping Identity Directory Server Configuration Reference* for more information).

Normal LDAP to LDAP synchronization topologies typically use dn as a correlation attribute. If dn is used as a correlation attribute only, the changelog-include-key-attribute property does not need to be set. However, if another attribute is used for correlation, this property must be set during the Sync Pipe configuration.

The LDAP change log attribute, ds-changelog-entry-key-attr-values, stores the attribute that is always included in a change log entry on every change for correlation purposes. In addition to regular attributes, virtual and operational attributes can be specified as entry keys.

To view an example, see the *Ping Identity Directory Server Administration Guide*.

changelog-max-before-after-values

The changelog-max-before-after-values property specifies the maximum number of "before and after" values (default 200) that should be stored for any changed attribute in the change log. Also, when enabled, it will add the ds-changelog-before-values and ds-changelog-after-values attributes to any change record that contains changes (for Modify and ModifyDN).

The main purpose of the changelog-max-before-after-values property is to ensure that an excessively large number of changes is not stored for multi-valued attributes. In most cases, the directory server’s schema defines a multi-valued attribute to be unlimited in an entry. For example, if a group entry whose member attribute references 10000 entries, the desire may be to not have all of the attributes if a new member added.

If either the ds-changelog-before-values or the ds-changelog-after-values attributes exceed the count set in the changelog-max-before-after-values property, the attribute values are no longer stored in a change record but its attribute name and number is stored in the ds-changelog-attr-exceeded-max-values-count attribute, which appears in the change record.

In addition to this property, set the use-reversible-form property to TRUE. This guarantees that sufficient information is stored in the change log for all operation types to be able to replay the operations at the destination. The create-sync-pipe-config tool configures these properties when it prepares the servers.

The changelog-max-before-after-values property configures the following change log attributes:

- ds-changelog-before-values – Captures all "before" values of a changed attribute. It will store up to the specified value in the changelog-max-before-after-values property (default 200).
- ds-changelog-after-values – Captures all "after" values of a changed attribute. It will store up to the specified value in the changelog-max-before-after-values property (default 200).
- ds-changelog-attr-exceeded-max-values-count – Stores the attribute names and number of before and after values on the change log entry after the maximum number of values (set by the changelog-max-before-after-values property) has been exceeded. This is a multi-valued attribute with the following format:

  ```
  attr=attributeName,beforeCount=200,afterCount=201
  ```

  where attributeName is the name of the attribute and the beforeCount and afterCount are the total number of values for that attribute before and after the change, respectively. In either case (before or after the change), if the number of values is exceeding the maximum, those values will not be stored.

LDAP change log for Notification and Standard Mode

Both Notification and Standard mode Sync Pipes can consume the same LDAP Change Log without affecting the other. Standard mode polls the change record in the change log for any modifications, fetches the full entries on the source and the destination, and then compares them for the specific changes. Notification mode gets the before and after values of a changed attribute to reconstruct an entry, and bypasses the fetch-and-compare phase. Both can consume the same LDAP Change Log with no performance loss or conflicts.

> **Note**
> If the configuration obtains the change log through a PingDirectory Proxy Server, the contents of the change log will not change as it is being read from the change logs on the directory server backend.

Implementing the Server Extension

Notification mode relies heavily on the server extension code to process and transmit the change using the required protocol and data formats needed for the client applications. Create the extension using the Server SDK, which provides the APIs to develop code for any destination endpoint type. The Server SDK’s documentation (Javadoc and examples) is delivered with the Server SDK built-in zip format. The SDK provides all of the necessary classes to extend the functionality of the PingDataSync Server without code changes to the core product. Once the server extension is in place, use other third-party libraries to transform the notification to any desired output format.

Consider the following when implementing the extension:

- **Use the manage-extension Tool** – Use the `manage-extension` tool in the `bin` directory or `bat` directory (Windows) to install or update the extension. See Managing Extensions for more information.
- **Review the Server SDK Package** – Review Server SDK documentation and examples before building and deploy a Java or Groovy extension.
- **Connection and Protocol Logic** – The Server SDK extension must manage the notification connection and protocol logic to the client applications.
- **Implementing Extensions** – Test the create methods, the delete methods, and the modify methods for each entry type. Update the configuration as needed. Finally, package the extensions for deployment. Logging levels can be increased to include more details.
- **Use the SyncOperation Type** – The `SyncOperation` class encapsulates everything to do with a given change. Objects of this type are used in all of the synchronization SDK extensions. See the Server SDK Javadoc for the `SyncOperation` class for information on the full set of methods.
- **Use the EndpointException Type** – The Sync Destination type offers an exception type called `EndpointException` to extend a standard Java exception and provide custom exceptions. There is also logic to handle LDAP exceptions, using the LDAP SDK.
- **About the PostStep result codes** – The `EndpointException` class uses `PostStep` result codes that are returned in the server extension:
  - `retry_operation_limited` – Retry a failed attempt up to the limit set by `max_operation_attempts`.
  - `retry_operation_unlimited` – Retry the operation an unlimited number of times until a success, abort, or `retried_operation_limited`. This should only be used when the destination endpoint is unavailable.
  - `abort_operation` – Abort the current operation without any additional processing.
- **Use the ServerContext class for logging** – The `ServerContext` class provides several logging methods which can be used to generate log messages and/or alerts from the scripted layer: `logMessage()`, `sendAlert()`, `debugCaught()`, `debugError()`, `debugInfo()`, `debugThrown()`, `debugVerbose()`, and `debugWarning()`. These are described in the Server SDK API Javadocs. Logging related to an individual `SyncOperation` should be done with the `SyncOperation#logInfo` and `SyncOperation#logError` methods.

- **Diagnosing Script Errors** – When a Groovy extension does not behave as expected, first look in the error log for stack traces. If `classLoader` errors are present, the script could be in the wrong location or may not have the correct package. Groovy checks for errors at runtime. Business logic errors must be systematically found by testing each operation. Make sure logger levels are set high enough to debug.

Configuring the Notification Sync Pipe

The following procedure configures a one-way Sync Pipe with a Directory Server as the Sync Source and a generic sync destination. The procedure uses the `create-sync-pipe-config` tool in interactive command-line mode and highlights the differences for configuring a Sync Pipe in notification mode.

Considerations for Configuring Sync Classes

When configuring a Sync Class for a Sync Pipe in notification mode, consider the following:

- Exclude any operational attributes from synchronizing to the destination so that its before and after values are not recorded in the change log. For example, the following attributes can be excluded: `creatorsName`, `createTimeStamp`, `ds-entry-unique-id`, `modifiersName`, and `modifyTimeStamp`. Filter the changes at the change log level instead of making the changes in the Sync Class to avoid extra configuration settings with the following:
    - Use the directory server's `changelog-exclude-attribute` property with (`+`) to exclude all operational attributes (`change-log-exclude-attribute:+`).
    - Configure a Sync Class that sets the `excluded-auto-mapped-source-attributes` property to each operational attribute to exclude from the synchronization process.
    - Use the directory server's `changelog-exclude-attribute` property to specify each operational attribute to exclude in the synchronization process. Set the configuration using the `dsconfig` tool on the directory server Change Log Backend menu. For example, `set changelog-exclude-attribute:modifiersName`.
- Use the `destination-create-only-attribute` advanced property on the Sync Class. This property sets the attributes to include on CREATE operations only.
- Use the `replace-all-attr-values` advanced property on the Sync Class. This property specifies whether to use the ADD and DELETE modification types (reversible), or the REPLACE modification type (non-reversible) for modifications to destination entries. If set to `true`, REPLACE is used.
- If targeting specific attributes that require higher performance throughput, consider implementing change log indexing. See Synchronizing Through Proxy Servers for more information.

Creating the Sync Pipe

The initial configuration steps show how to set up a single Sync Pipe from a directory server instance to a generic Sync Destination.

Before starting:

- Place any third-party libraries in the `<server-root>/lib/extensions` folder.
- Implement a server extension for any custom endpoints and place it in the appropriate directory.

1. If necessary, start the PingDataSync Server:

   ```
   $ bin/start-server
   ```

2. Run the `create-sync-pipe-config` tool.

   ```
   $ bin/create-sync-pipe-config
   ```

3. At the Initial Synchronization Configuration Tool prompt, press **Enter** to continue.
4. On the Synchronization Mode menu, select the option for notification mode.
5. On the Synchronization Directory menu, enter the option to create a one-way Sync Pipe in notification mode from directory to a generic client application.

Configuring the Sync Source

1. On the Source Endpoint Type menu, enter the option for the Sync Source type.
2. Choose a pre-existing Sync Source, or create a new sync source.
3. Enter a name for the Source Endpoint and a name for the Sync Source.
4. Enter the base DN for the directory server used for LDAP searches, such as `dc=example,dc=com`, and press **Enter** to return to the menu. If entering more than one base DN, make sure they do not overlap.
5. On the Server Security menu, select the type of communication that the PingDataSync Server will use with endpoint servers.
6. Enter the host and port of the first Source Endpoint server. The Sync Source can specify a single server or multiple servers in a replicated topology. The PingDataSync Server contacts this first server if it is available, then contacts the next highest priority server if the first server is unavailable. The server tests the connection.
7. On the Sync User Account menu, enter the DN of the sync user account and password, or press **Enter** to accept the default, `cn=Sync User,cn=Root DNs,cn=config`. This account allows the PingDataSync Server to access the source endpoint server.

Configure the Destination Endpoint Server

1. On the Destination Endpoint Type menu, select the type of data store on the endpoint server. In this example, select the option for Custom.
2. Enter a name for the Destination Endpoint and a name for the Sync Destination.
3. On the Notifications Setup menu, select the language (Java or Groovy) used to write the server extension.
4. Enter the fully qualified name of the Server SDK extension that implements the abstract class. A Java, extension should reside in the `/lib/extensions` directory. A Groovy script should reside in the `/lib/groovy-scripted-extensions` directory.
5. Configure any user-defined arguments needed by the server extension. Typically, these are connection arguments, which are defined by the extension itself. The values are then entered here and stored in the server configuration.
6. Configure the maximum number of before and after values for all changed attributes. Notification mode requires this. Set the cap to something well above the maximum number of values that any synchronized attribute will have. If this cap is exceeded, the PingDataSync Server will issue an alert. For this example, we accept the default value of 200.

   ```
   Enter a value for the max changelog before/after values,
   or -1 for no limit [200]:
   ```

7. Configure any key attributes in the change log that should be included in every notification. These attributes can be used to find the destination entry corresponding to the source entry, and will be present whether or not the attributes changed. Later, any attributes used in a Sync Class include-filter should also be configured as key attributes in the Sync Class.
8. In both standard and notification modes, the Sync Pipe processes the changes concurrently with multiple threads. If changes must be applied strictly in order, the number of Sync Pipe worker threads will be reduced to 1. This will limit the maximum throughput of the Sync Pipe.

The rest of the configuration steps follow the same process as a standard synchronization mode Sync Pipe. See About the Sync User Account for more information.

Access control filtering on the Sync Pipe

The PingDataSync Server provides an advanced Sync Pipe configuration property, `filter-changes-by-user`, that performs access control filtering on a changelog entry for a specific user.

Since the changelog entry contains data from the target entry, the access controls filter out attributes that the user does not have the privileges to see before it is returned. For example, values in the changes, `ds-changelog-before-values`, `ds-changelog-after-values`, `ds-changelog-entry-key-attr-values`, and `deletedEntryAttrs` are filtered out through access control instructions.

> **Note**
> This property is only available for notification mode and can be configured using the `create-sync-pipe-config` or the `dsconfig` tool.

The source server must be an Ping Identity Directory Server or Alcatel-Lucent 8661 Directory Server, or an Ping Identity PingDirectory Proxy Server or Alcatel-Lucent 8661 Directory Proxy Server that points to an Ping Identity Directory Server or Alcatel-Lucent 8661 Directory Server.

Considerations for access control filtering

- The directory server will not return the changelog entry if the user is not allowed to see the target entry.
- The directory server strips out any attributes that the user is not allowed to see.
- If no changes are left in the entry, no changelog entry will be returned.
- If only some attributes are stripped out, the changelog entry will be returned.
- Access control filtering on a specific attribute value is not supported. Either all attribute values are returned or none.
- If a sensitive attribute policy is used to filter attributes when a client normally accesses the directory server, this policy will not be taken into consideration during notifications since the Sync User is always connecting using the same method. Configure access controls to filter out attributes, not based on the type of connection made to the server, but based on who is accessing the data. The `filter-changes-by-user` property will be able to evaluate if that person should have access to these attributes.

Configure the Sync Pipe to filter changes by access control instructions

1. Set the `filter-changes-by-user` property to filter changes based on access controls for a specific user.

   ```
   $ bin/dsconfig set-sync-pipe-prop \
     --pipe-name "Notifications Sync Pipe" \
     --set "filter-changes-by-user:uid=admin,dc=example,dc=com"
   ```

2. On the source directory server, set the `report-excluded-changelog-attributes` property to include the names of users that have been removed through access control filtering. This will allow the PingDataSync Server to warn about attributes that were supposed to be synchronized but were filtered out. This step is recommended but not required.

```
$ bin/dsconfig set-backend-prop \
--backend-name "changelog" \
  --set "report-excluded-changelog-attributes:attribute-names"
```

Note

The PingDataSync Server only uses the `attribute-names` setting for the Directory Server's `report-excluded-changelog-attributes` property. It does not use the `attribute-counts` setting for the property.

Chapter 8: Configure synchronization with SCIM

The PingDataSync Server provides data synchronization between directory servers or proxy servers and System for Cross-domain Identity Management (SCIM) applications over HTTP. Synchronization can be done with custom SCIM applications, or with the PingData Directory Server and PingDirectory Proxy Server configured as SCIM servers using the SCIM extension.

Topics include:

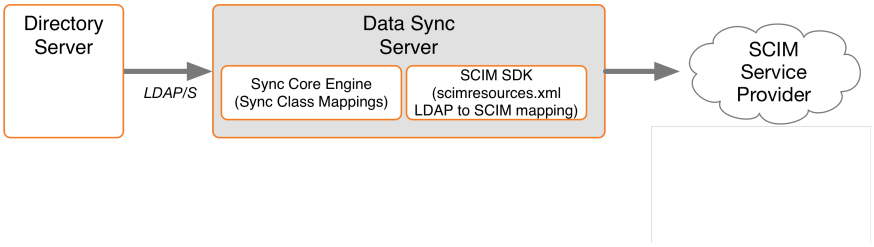
Synchronize with a SCIM Sync Destination overview

The SCIM protocol is designed to make managing user identity in cloud-based applications and services easier. SCIM enables provisioning identities, groups, and passwords to, from, and between clouds. The PingDataSync Server can be configured to synchronize with SCIM service providers.

> **Note**
> Both the Ping Identity Directory Server and PingDirectory Proxy Server can be configured to be SCIM servers using the SCIM HTTP Servlet Extension.

The PingDataSync Server is LDAP-centric and operates on LDAP attributes. The SCIM Sync Destination server component acts as a translation layer between a SCIM service provider's schema and an LDAP representation of the entries. While the PingDataSync Server is LDAP-centric and typically at least one endpoint is an LDAP Directory Server, this is not a strict requirement. For example, a JDBC to SCIM sync pipe can be configured.

The PingDataSync Server contains sync classes that define how source and destination entries are correlated. The SCIM Sync Destination contains its own mapping layer, based on `scimresources.xml` that maps LDAP schema to and from SCIM.


Directory Server
LDAP/S
Data Sync Server
Sync Core Engine (Sync Class Mappings)
SCIM SDK (scimresources.xml LDAP to SCIM mapping)
SCIM Service Provider

Synchronizing with a SCIM Sync Destination

> **Note**
> The PingDataSync Server can only use SCIM as a Sync Destination. There is no mechanism in the SCIM protocol for detecting changes, so it cannot be used as a Sync Source.

SCIM destination configuration objects

The `SCIMSyncDestination` object defines a SCIM service provider Sync Pipe destination that is accessible over HTTP through the SCIM protocol. It is configured with the following properties:

- `server` – Specifies the names of the SCIM External Servers that are used as the destination of synchronization.

- `resource-mapping-file` – Specifies the path to the `scim-resources.xml` file, a configuration file that defines the SCIM schema and maps it to the LDAP schema. This file is located in `<server root>/config/scim-resources.xml` by default. This file can be customized to define and expose deployment-specific resources.
- `rename-policy` – Specifies how to handle the rename of a SCIM resource.

The SCIM Sync Destination object is based on the SCIM SDK. Before configuring a SCIM destination, review the following documents on the Simple Cloud web site:

- SCIM Core Schema
- SCIM REST API

Considerations for synchronizing to a SCIM destination

When configuring an LDAP to SCIM Sync Pipe, consider the following:

- **Use `scim-resources.xml` for Attribute and DN Mappings** – There are two layers of mapping: once at the Sync Class level and again at the SCIM Sync Destination level in the `scim-resources.xml` file. To reduce complexity, do all possible mappings in the `scim-resources.xml` file.
- **Avoid Groups Unless the SCIM ID is DN Based** – Group synchronization is supported if the SCIM ID is based on the DN. If the SCIM ID is not the DN itself, it must be one of the components of the RDN, meaning that the DNs of group members must contain the necessary attribute.
- **SCIM Modifies Entries Using PUT** – The SCIM Sync Destination modifies entries using the full HTTP PUT method. For every modify, SCIM replaces the entire resource with the updated resource. For information about the implications of this on password updates, see Password Considerations with SCIM.

Renaming a SCIM resource

The SCIM protocol does not support changes that require the SCIM resource to be renamed, such as a MODDN operation. Instead, when a change is detected to an attribute value that is used as part of the SCIM ID attribute, the PingDataSync Server handles it in one of the following ways:

- Deletes the specified SCIM resource and then adds the new resource with the new SCIM ID.
- Adds the new resource with the new SCIM ID and then deletes the old resource.
- Skips the rename portion of the change. If renames are expected on the source endpoint, a careful set of destination-correlation attributes should be chosen so that the destination can still be found after it is renamed on the source.

Configure this by setting the `rename-policy` property of the SCIM Sync Destination.

Password considerations with SCIM

Because the SCIM sync destination modifies entries using a full PUT method, special considerations need to be made for password attributes. An Ping Identity SCIM Server allows password attributes to be omitted from a change when they have not been modified by an operation. This prevents passwords from inadvertently being overwritten during the PUT operation, which does not include the password attribute. Ideally, other SCIM service providers will not wipe a password because a PUT request does not contain it. Check with the SCIM vendor to confirm this behavior before starting a SCIM sync pipe.

Configure synchronization with SCIM

Configure synchronization with SCIM using the `create-sync-pipe-config` utility and the `dsconfig` command. Configuring synchronization between an LDAP server and a SCIM service provider includes the following:

- Configure one external server for every physical endpoint.
- Configure the Sync Source server and designate the external servers that correspond to the source server.
- Configure the Sync Destination server and designate the external servers that correspond to the SCIM sync destination.
- Configure the LDAP to SCIM Sync Pipe.
- Configure the Sync Classes. Each Sync Class represents a type of entry that needs to be synchronized. When specifying a Sync Class for synchronization with a SCIM service provider, avoid including attribute and DN mappings. Instead use the Sync Class to specify the operations to synchronize and which correlation attributes to use.
- Set the evaluation order for the Sync Classes to define the processing precedence for each class.
- Configure the `scim-resources.xml` file. If possible, change the `<resourceIDMapping>` element(s) to use whatever the SCIM Service Provider uses as the SCIM ID.
- Set Up Communication for each External Server. Run `prepare-endpoint-server` once for every LDAP external server that is part of the Sync Source.
- Use `realtime-sync` to start the Sync Pipe.

Configure the external servers

Perform the following to configure an external server for each host in the deployment:

1. Configure an Directory Server as an external server, which will later be configured as a Sync Source. On the PingDataSync Server, run the following `dsconfig` command:

```
$ bin/dsconfig create-external-server \
  --server-name source-ds \
  --type ping-identity-ds \
  --set server-host-name:ds1.example.com \
  --set server-port:636 \
  --set "bind-dn:cn=Directory Manager" \
  --set password:secret \
  --set connection-security:ssl \
  --set key-manager-provider:Null \
  --set trust-manager-provider:JKS
```

2. Configure the SCIM server as an external server, which will later be configured as a Sync Destination. The `scim-service-url` property specifies the complete URL used to access the SCIM service provider. The `user-name` property specifies the account used to connect to the SCIM service provider. By default, the value is `cn=Sync User,cn=Root DNs,cn=config`. Some SCIM service providers may not have the user name in DN format.

```
$ bin/dsconfig create-external-server \
  --server-name scim \
  --type scim \
  --set scim-service-url:https://scim1.example.com:8443 \
  --set "user-name:cn=Sync User,cn=Root DNs,cn=config" \
  --set password:secret \
  --set connection-security:ssl \
  --set hostname-verification-method:strict \
  --set trust-manager-provider:JKS
```

Configure the Directory Server Sync Source

Configure the Sync Source for the synchronization network. More than one external server can be configured to act as the Sync Source for failover purposes. If the source is an Directory Server, also configure the following items:

- Enable the changelog password encryption plug-in on any directory server that will receive password modifications. This plugin intercepts password modifications, encrypts the password, and adds an encrypted attribute to the change log entry.
- Configure the `changelog-deleted-entry-include-attribute` property on the changelog backend, so that the PingDataSync Server can record which attributes were removed during a DELETE operation.

Perform the following steps to configure the Sync Source:

1. Run `dsconfig` to configure the external server as the Sync Source. Based on the previous example where the Directory Server was configured as `source-ds`, run the following command:

```
$ bin/dsconfig create-sync-source --source-name source \
  --type ping-identity \
  --set base-dn:dc=example,dc=com \
  --set server:source-ds \
  --set use-changelog-batch-request:true
```

2. Enable the change log password encryption plug-in on any server that will receive password modifications. The encryption key can be copied from the output, if displayed, or accessed from the `<server-root>/bin/sync-pipe-cfg.txt` file, if the `create-sync-pipe-config` tool was used to create the sync pipe.

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Changelog Password Encryption" \
  --set enabled:true \
  --set changelog-password-encryption-key:<key>
```

3. On the PingDataSync Server, set the decryption key used to decrypt the user password value in the change log entries. The key allows the user password to be synchronized to other servers that do not use the same password storage scheme.

```
$ bin/dsconfig set-global-sync-configuration-prop \
  --set changelog-password-decryption-key:ej5u9e39pq-68
```

4. Configure the `changelog-deleted-entry-include-attribute` property on the changelog backend.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set changelog-deleted-entry-include-attribute:objectClass
```

Configure the SCIM Sync Destination

Configure the SCIM Sync Destination to synchronize data with a SCIM service provider. Run the `dsconfig` command:

```
$ bin/dsconfig create-sync-destination \
  --destination-name scim \
  --type scim \
  --set server:scim
```

Configure the Sync Pipe, Sync Classes, and evaluation order

Configure a Sync Pipe for LDAP to SCIM synchronization, create Sync Classes for the Sync Pipe, and set the evaluation order index for the Sync Classes.

Note
The Synchronization mode must be set to Standard. Notification Mode cannot be used with SCIM.

1. Once the source and destination endpoints are configured, configure the Sync Pipe for LDAP to SCIM synchronization. Run the `dsconfig` command to configure an LDAP-to-SCIM Sync Pipe:

```
$ bin/dsconfig create-sync-pipe \
  --pipe-name ldap-to-scim \
  --set sync-source:source \
  --set sync-destination:scim
```

2. The next set of steps define three Sync Classes. The first Sync Class is used to match user entries in the Sync Source. The second class is used to match group entries. The third class is a DEFAULT class that is used to match all other entries.

   Run the `dsconfig` command to create the first Sync Class and set the Sync Pipe Name and Sync Class name:

   ```
   $ bin/dsconfig create-sync-class \
     --pipe-name ldap-to-scim \
     --class-name user
   ```

3. Use `dsconfig` to set the base DN and filter for this Sync Class. The `include-base-dn` property specifies the base DN in the source, which is `ou=people,dc=example,dc=com` by default. This Sync Class is invoked only for changes at the `ou=people` level. The `include-filter` property specifies an LDAP filter that tells the PingDataSync Server to include `inetOrgPerson` entries as user entries. The `destination-correlation-attributes` specifies LDAP attributes that allow the PingDataSync Server to find the destination resource on the SCIM server. The value of this property will vary. See Identifying a SCIM Resource at the Destination Server for details.

   ```
   $ bin/dsconfig set-sync-class-prop \
     --pipe-name ldap-to-scim \
     --class-name user \
     --add include-base-dn:ou=people,dc=example,dc=com \
     --add "include-filter:(objectClass=inetOrgPerson)" \
     --set destination-correlation-attributes:externalId
   ```

4. Create the second Sync Class, which is used to match group entries.

   ```
   $ bin/dsconfig create-sync-class \
     --pipe-name ldap-to-scim \
     --class-name group
   ```

5. For the second Sync Class, set the base DN and the filters to match the group entries.

   ```
   $ bin/dsconfig set-sync-class-prop \
     --pipe-name ldap-to-scim \
     --class-name group \
     --add include-base-dn:ou=groups,dc=example,dc=com \
     --add "include-filter:(|(objectClass=groupOfEntries)\
       (objectClass=groupOfNames)(objectClass=groupOfUniqueNames)\
       (objectClass=groupOfURLs))"
   ```

6. For the third Sync Class, create a DEFAULT Sync Class that is used to match all other entries. To synchronize changes from only user and group entries, set `synchronize-creates`, `synchronize-modifies`, and `synchronize-delete` to `false`.

   ```
   $ bin/dsconfig create-sync-class \
     --pipe-name ldap-to-scim \
   ```

```
  --class-name DEFAULT \
  --set evaluation-order-index:99999 \
  --set synchronize-creates:false \
  --set synchronize-modifies:false \
  --set synchronize-deletes:false
```

7. After all of the Sync Classes needed by the Sync Pipe are configured, set the evaluation order index for each Sync Class. Classes with a lower number are evaluated first. Run `dsconfig` to set the evaluation order index for the Sync Class. The actual number depends on the deployment.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name ldap-to-scim \
  --class-name user \
  --set evaluation-order-index:100
```

Configure communication with the source server(s)

Configure communication between the PingDataSync Server and the LDAP source servers with the `prepare-endpoint-server` tool. If user accounts do not exist, this tool creates the appropriate user account and its privileges. Also, because the source is an Directory Server, this tool enables the change log.

> **Note**
> The `prepare-endpoint-server` tool can only be used on LDAP directory servers. For the SCIM Server, manually create a sync user entry.

Run the `prepare-endpoint-server` command to setup communication between the PingDataSync Server and the source server(s). The tool will prompt for the bind DN and password to create the user account and enable the change log.

```
$ bin/prepare-endpoint-server \
  --hostname ds1.example.com \
  --port 636 \
  --useSSL \
  --trustAll \
  --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
  --syncServerBindPassword "password" \
  --baseDN "dc=example,dc=com" \
  --isSource
```

Start the Sync Pipe

The `realtime-sync` tool sets a specific starting point for real-time synchronization, so that changes made before the current time are ignored.

1. Run the `realtime-sync` tool to set the startpoint for the Sync Source.

```
$ bin/realtime-sync set-startpoint \
  --end-of-changelog \
  --pipe-name ldap-to-scim
```

2. When ready to start synchronization, run the following command:

   ```
   $ bin/realtime-sync start \
     --pipe-name ldap-to-scim \
     --no-prompt
   ```

Map LDAP schema to SCIM resource schema

The resources configuration file is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for standard SCIM Users and Groups resources, and mappings to standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes. It is installed with the Directory Server. This file can be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

The default configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. In all cases, this must be changed to match the attribute that the destination SCIM service provider is using for its SCIM resource ID. For example, if the destination uses the value of the `uid` attribute, modify the `scim-resources.xml` file to change the `resourceIDMapping` as follows:

```
<resourceIDMapping ldapAttribute="uid" />
```

Ideally, this would be an attribute that exists on the source LDAP entry. If not, the PingDataSync Server can construct it using a Constructed Attribute Mapping. For example, the SCIM service provider used the first and last initials of the user, concatenated with the employee ID (given by the `eid` attribute) as the SCIM resource ID. In this case, an attribute mapping would be constructed as follows:

```
$ dsconfig create-attribute-mapping \
  --map-name MyAttrMap \
  --mapping-name scimID \
  --type constructed \
  --set 'value-pattern:{givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid}'
```

This creates an attribute called `scimID` on the mapped entry when processed by the Sync engine. For example, if the user's name was John Smith, with employee ID 12345, then the `scimID` would be `js12345`. Once this is done,configure the `scim-resources.xml` file as follows:

```
<resourceIDMapping ldapAttribute="scimID" />
```

This will cause it to pull out the constructed `scimID` value from the entry and use that at the SCIM resource ID when making requests to the service provider.

> **Note**
> Constructed attribute mappings support multivalued source attributes for conditional (using the `conditional-value-pattern` property) and non-conditional (using the `value-pattern` property) value patterns. Only one of the source attributes that contribute to a given value pattern can be multivalued.

For any given SCIM resource endpoint, only one <LDAPAdd> template can be defined, and only one <LDAPSearch> element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. If using the SCIM HTTP Servlet Extension for the Directory Server, this can be implemented in many ways, such as:

- Create multiple SCIM servlets, each with a unique resources.xml configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique resources.xml configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). Make sure that any attributes that contain binary data are declared using dataType=binary in the scim-resources.xml file. When using the Identity Access API, make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes that contain binary data. This instructs the server to base64-encode the data before returning it to clients.

If attributes that are not declared as binary in the schema and contain binary data (or just data that is invalid in XML), the server will check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values that include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the scimresources.xml file.

The <resource> element

A resource element has the following XML attributes:

- **schema**: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as urn:scim:schemas:core:1.0. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- **name**: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- **mapping**: a custom Java class that provides the logic for the resource mapper. This class must extend the com.unboundid.scim.ldap.ResourceMapper class.

A resource element contains the following XML elements in sequence:

- **description**: a required element describing the resource.
- **endpoint**: a required element specifying the endpoint to access the resource using the SCIM REST API.
- **LDAPSearchRef**: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- **LDAPAdd**: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- **attribute**: one or more elements specifying the SCIM attributes for the resource.

The <attribute> element

An `attribute` element has the following XML attributes:

- **schema**: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- **name**: a required attribute specifying the name of the SCIM attribute.
- **readOnly**: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is `false`.
- **required**: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is `false`.

An `attribute` element contains the following XML elements in sequence:

- **description**: a required element describing the attribute. Then just one of the following elements:
- **simple**: specifies a simple, singular SCIM attribute.
- **complex**: specifies a complex, singular SCIM attribute.
- **simpleMultiValued**: specifies a simple, multi-valued SCIM attribute.
- **complexMultiValued**: specifies a complex, multi-valued SCIM attribute.

The <simple> element

A `simple` element has the following XML attributes:

- **dataType**: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `decimal`, `integer`, and `string`.
- **caseExact**: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is `false`.

A `simple` element contains the following XML elements in sequence:

- **mapping**: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

The <complex> element

The `complex` element does not have any XML attributes. It contains the following XML element:

- **subAttribute**: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard type, primary, and display sub-attributes do not need to be specified.

The <simpleMultiValued> element

A `simpleMultiValued` element has the following XML attributes:

- **childName**: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- **dataType**: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, and `string`.
- **caseExact**: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is `false`.

A `simpleMultiValued` element contains the following XML elements in sequence:

- **canonicalValue**: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- **mapping**: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

The <complexMultiValued> element

A `complexMultiValued` element has the following XML attributes:

- **tag**: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard `addresses` SCIM attribute is `address`.

A `complexMultiValued` element contains the following XML elements in sequence:

- **subAttribute**: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.
- **canonicalValue**: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

The <subAttribute> element

A `subAttribute` element has the following XML attributes:

- **name**: a required element specifying the name of the sub-attribute.
- **readOnly**: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is `false`.
- **required**: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is `false`.
- **dataType**: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, and `string`.
- **caseExact**: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is `false`.

A `subAttribute` element contains the following XML elements in sequence:

- **description**: a required element describing the sub-attribute.
- **mapping**: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the `complexMultiValued` element.

The <canonicalValue> element

A `canonicalValue` element has the following XML attributes:

- **name**: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML elements in sequence:

- **subMapping**: an optional element specifying mappings for one or more of the subattributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

The <mapping> element

A `mapping` element has the following XML attributes:

- **ldapAttribute**: a required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- **transform**: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP, and LDAP to SCIM. The available transformations are described in Mapping LDAP Schema to SCIM Resource Schema.

The <subMapping> element

A `subMapping` element has the following XML attributes:

- **name**: a required element specifying the name of the sub-attribute that is mapped.
- **ldapAttribute**: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- **transform**: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. Available transformations are described in Mapping LDAP Schema to SCIM Resource Schema.

The <LDAPSearch> element

A `LDAPSearch` element has the following XML attributes:

- **baseDN**: a required element specifying the LDAP search base DN to be used when querying for the SCIM resource.
- **filter**: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- **resourceIDMapping**: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN.

Note

The `LDAPSearch` element can be added as a top-level element outside of any `<Resource>` elements, and then referenced within them with an `ID` attribute.

The <resourceIDMapping> element

A `resourceIDMapping` element has the following XML attributes:

- **ldapAttribute**: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- **createdBy**: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `<scim-consumer>`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or directory, (as would be the case if the mapped LDAP attribute is `entryUUID`).

If the LDAP attribute value is not listed as destination correlation attribute, this setting is not used by the PingDataSync Server.

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

The <LDAPAdd> element

A `LDAPAdd` element has the following XML attributes:

- **DNTemplate**: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- **fixedAttribute**: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

The <fixedAttribute> element

A `fixedAttribute` element has the following XML attributes:

- **ldapAttribute**: a required attribute specifying the name of the LDAP attribute for the fixed values.
- **onConflict**: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The default value `merge` indicates that the fixed values should be merged with the existing values. The value `overwrite` indicates that the existing values are to be overwritten by the fixed values. The value `preserve` indicates that no changes should be made.

A fixedAttribute element contains the following XML element:

- **fixedValue**: one or more elements specifying the fixed LDAP values.

Identify a SCIM resource at the destination

When a SCIM Sync Destination needs to synchronize a change to a SCIM resource on the destination SCIM server, it must first fetch the destination resource. If the destination resource ID is known, the resource will be retrieved by its ID. If not, a search is performed using the mapped destination correlation attributes. Configuring this requires coordination between the Sync Class and the scim-resources.xml mapping file.

The scim-resources.xml mapping file treats the value of the <resourceIDMapping> element's ldapAttribute attribute as the SCIM ID of the source entry. If this value is also listed as a value of the Sync Class's destination-correlation-attributes property, then the value of this LDAP attribute is used as the SCIM ID of the destination resource.

If no value of destination-correlation-attributes matches the <resourceIDMapping> element's ldapAttribute attribute, the SCIM ID of the destination resource is considered unknown. In this case, the SCIM Sync Destination treats the values of destination-correlation-attributes as search terms, using them to construct a filter for finding the destination resource. Each value of destination-correlation-attributes will be mapped to a corresponding SCIM attribute name, and equality matches will be used in the resulting filter.

If the ldapAttribute value is not listed as a destination correlation attribute, this setting is not used by the PingDataSync Server.

The following table illustrates an LDAPSearch element that contains a resourceIDMapping element:

Identifying a SCIM Resource

Method for Retrieving SCIM Resource	Condition	Example Condition	Example Request
Retrieve resource directly	Used if a destination-correlation-attribute value matches the <resourceIDMapping> ldapAttribute value.	destination-correlation-attribute=mail,uid;<resourceIDMapping ldapAttribute="mail" createdBy= "directory"/>	GET scim/Users/ person@example.com
Retrieve resource using search	Used if no destination-correlation-attribute value matches the <resourceIDMappin	destination-correlation-attribute=mail,uid;<resourceIDMapping ldapAttribute="entryUUID"createdBy= "directory"/>	GET /scim/Users? filter=emails+eq +"person@example.com" and+userName +eq"person"

Identifying a SCIM Resource

Method for Retrieving SCIM Resource	Condition	Example Condition	Example Request
	g> `ldapAttribute` value.		

The unique ID of a destination SCIM resource will most likely be unknown, and the search method will need to be used. However, not all SCIM service providers support the use of filters. Therefore, not all SCIM service providers may be usable as SCIM Sync Destinations.

Chapter 9: Manage logging, alerts, and alarms

Each PingData server supports extensive logging features to track all aspects of the PingData topology.

Topics include:

Logs and log publishers

Synchronization logs and messages

Create a new log publisher

Configure log signing

Configure log retention and rotation policies

Configure log listeners

System alarms, alerts, and gauges

Test alerts and alarms

The status tool

Sync-specific status

Monitor the PingDataSync Server

Logs and Log Publishers

PingData servers support different types of log publishers that can be used to provide the monitoring information for operations, access, debug, and error messages that occur during normal server processing. The server provides a standard set of default log files as well as mechanisms to configure custom log publishers with their own log rotation and retention policies.

Types of Log Publishers

Several types of log publishers can be used to log processing information about the server, including:

- **Audit loggers** – provide information about actions that occur within the server. Specifically, this type of log records all changes applied, detected or failed; dropped operations that were not completed; changes dropped due to being out of scope, or no changes needed for an operation. The log also shows the entries that were involved in a process.
- **Error loggers** – provide information about warnings, errors, or significant events that occur within the server.
- **Debug loggers** – provide detailed information about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database.
- **Access loggers** – provide information about LDAP operations processed within the server. This log only applies to operations performed in the server. This includes configuration changes, searches of monitor data, and bind operations for authenticating administrators using the command-line tools and the Administrative Console.

View the list of log publishers

View the list of log publishers on each server using the `dsconfig` tool:

```
$ bin/dsconfig list-log-publishers

Log Publisher                       : Type              : enabled
-----------------------------------:-------------------:--------
Debug ACI Logger                    : debug-access      : false
Expensive Operations Access Logger : file-based-access : false
Failed Operations Access Logger     : file-based-access : true
File-Based Access Logger            : file-based-access : true
File-Based Audit Logger             : file-based-audit  : false
File-Based Debug Logger             : file-based-debug  : false
File-Based Error Logger             : file-based-error  : true
Replication Repair Logger           : file-based-error  : true
```

Log compression

PingData servers support the ability to compress log files as they are written. Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled when the logger is created. Compression cannot be turned on or off once the logger is configured. If the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append it to the previous file.

Compression is performed using the standard gzip algorithm. Because it can be useful to have an amount of uncompressed log data for troubleshooting, having a second logger defined that does not use compression may be desired.

Configure compression by setting the `compression-mechanism` property to have the value of `gzip` when creating a new logger. See Creating a New Log Publisher for details.

Configure log file encryption

The server supports the ability to encrypt log files as they are written. The encrypt-log configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an .encrypted extension (and if both encryption and compression are enabled, the extension will be `.gz.encrypted`). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The `encrypt-file` tool (or custom code, using the LDAP SDK's `com.unboundid.util.PassphraseEncryptedInputStream`) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the `encryption-settings create` command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the `encryption-settings-definition-id` property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the `rotate-log` tool to force the file to be rotated before attempting to examine it.

The following commands can be used to set log file encryption:

1. Use `dsconfig` to enable encryption for a Log Publisher. In this example, the File-basedAccess Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

   ```
   $ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted
   Access" \
     --type file-based-access \
     --set enabled:true \
     --set compression-mechanism:gzip \
     --set encryption-settings-definition-
   id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
     --set encrypt-log:true \
     --set log-file:logs/encrypted-access \
     --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
     --set "rotation-policy:Size Limit Rotation Policy" \
      --set "retention-policy:File Count Retention Policy" \
     --set "retention-policy:Free Disk Space Retention Policy" \
      --set "retention-policy:Size Limit Retention Policy"
   ```

2. To decrypt and decompress the file:

   ```
   $ bin/encrypt-file --decrypt \
     --decompress-input \
     --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
     --output-file decrypted-access
   Initializing the server's encryption framework...DoneWriting decrypted
   data to file '/ds/PingDirectory/decrypted-access' using akey generated
   from encryption settings definition
   '332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'Success
   fully wrote 123,456,789 bytes of decrypted data
   ```

Synchronization logs and messages

The PingDataSync Server provides a standard set of default log files to monitor the server activity. View this set of logs in the `<server-root>/logs` directory. The following default log files are available.

PingDataSync Server Logs

Log File	Description
access	File-based Access Log that records LDAP operations processed by the PingDataSync Server. Access log records can be used to provide information about problems during operation processing and provide information about the time required to process each operation.
config-audit.log	Records information about changes made to the server configuration in a format that can be replayed using the `dsconfig` tool.
errors	File-based Error Log that provides information about warnings, errors, and significant events that are not errors but occur during server processing.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to

Log File	Description
	server.out.
server.pid	Stores the server's process ID.
server.status	Stores the timestamp, a status code, and an optional message providing additional information on the server status.
setup.log	Records messages that occur during the initial server configuration with the `setup` command.
sync	File-based Sync Log that records synchronization operations processed by the server. Specifically, the log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization.
sync-pipe-cfg.txt	Records the configuration changes used with the `bin/create-sync-pipe-config` tool. The file is placed wherever the tool is run. Typically, this is in `<server-root>` or in the `bin` directory.
tools	Holds logs for long running utilities. Current and previous copies of the log are present in the directory.
update.log	Records messages that occur during a server upgrade.

Sync log message types

The PingDataSync Server logs certain types of log messages with the sync log. Message types can be included or excluded from the logger, or added to a custom log publisher.

Sync Log Message Types

Message Type	Description
change-applied	Default summary message. Logged each time a change is applied successfully.
change-detected	Default summary message. Logged each time a change is detected.
change-failed-detailed	Default detail message. Logged when a change cannot be applied. It includes the reason for the failure and details about the change that can be used to manually repair the failure.
dropped-op-type-not-synchronized	Default summary message. Logged when a change is dropped because the operation type (for example, ADD) is not synchronized for the matching Sync Class.
dropped-out-of-scope	Default summary message. Logged when a change is dropped because it does not match any Sync Class.
no-change-needed	Default summary message. Logged each time a change is dropped because the modified source entry is already synchronized with the destination entry.
change-detected-detailed	Optional detail message. Logged each time a change is detected. It includes attribute values for added and modified entries. This information is useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
entry-mapping-details	Optional detail message. Logged each time a source entry (attributes and DN) are mapped to a destination entry. This information is useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
change-applied-detailed	Optional detail message. Logged each time a change is applied. It includes attribute values for added and modified entries. This information is useful for diagnosing

Message Type	Description
	problems, but it causes log files to grow faster, which impacts performance.
change-failed	Optional summary message. Logged when a change cannot be applied. It includes the reason for the failure but not enough information to manually repair the failure.
intermediate-failure	Optional summary message. Logged each time an attempt to apply a change fails. Note that a subsequent retry of applying the change might succeed.

Create a new log publisher

PingData servers provide customization options to create log publishers with the `dsconfig` command.

After creating a new log publisher, configure the log retention and rotation policies. For more information, see Configure log rotation and Configure log retention.

1. Use the `dsconfig` command to create and configure the new log publisher. (If using `dsconfig` in interactive mode, log publishers are created and manged under the Log Publisher menu.) The following example shows how to create a logger that only logs disconnect operations.

   ```
   $ bin/dsconfig create-log-publisher \
     --type file-based-access --publisher-name "Disconnect Logger" \
     --set enabled:true \
     --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
     --set "rotation-policy:Size Limit Rotation Policy" \
     --set "retention-policy:File Count Retention Policy" \
     --set log-connects:false \
     --set log-requests:false --set log-results:false \
     --set log-file:logs/disconnect.log
   ```

   To configure compression on the logger, add the following option to the previous command:

   ```
   --set compression-mechanism: gzip
   ```

   Compression cannot be disabled or turned off once configured for the logger. Determine logging requirements before configuring this option.

2. View log publishers with the following command:

   ```
   $ bin/dsconfig list-log-publishers
   ```

Configuring log signing

PingData servers support the ability to cryptographically sign a log to ensure that it has not been modified. For example, financial institutions require tamper-proof audit logs files to ensure that transactions can be properly validated and ensure that they have not been modified by a third-party entity or internally by an unauthorized person.

When enabling signing for a logger that already exists, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the `validate-file-signature` tool provided in the `bin` directory of the server (or the `bat` directory on Windows systems). Once this property is enabled, disable and then re-enable the log publisher for the changes to take effect. Perform the following steps to configure log signing:

1. Use `dsconfig` to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:false
```

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true
```

3. To validate a signed file, use the `validate-file-signature` tool to check if a signed file has been altered.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

If any validations errors occur, a message displays that is similar to this:

```
One or more signature validation errors were encountered while validating
the contents of file 'logs/audit':
* The end of the input stream was encountered without encountering the end
of an active signature block. The contents of this signed block cannot be
trusted because the signature cannot be verified
```

Configure log retention and log rotation policies

PingData servers enable configuring log rotation and log retention policies.

Log Retention – When any retention limit is reached, the server removes the oldest archived log prior to creating a new log. Log retention is only effective if a log rotation policy is in place. A new log publisher must have at least one log retention policy configured. The following policies are available:

- **File Count Retention Policy** – Sets the number of log files for the server to retain. The default file count is 10 logs. If the file count is set to 1, the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy** – Sets the minimum amount of free disk space. The default free disk space is 500 MB.
- **Size Limit Retention Policy** – Sets the maximum size of the combined archived logs. The default size limit is 500 MB.
- **Custom Retention Policy** – Create a new retention policy that meets the server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy** – Used in a rare event that does not require log deletion.

Log Rotation – When a rotation limit is reached, the server rotates the current log and starts a new log. A new log publisher must have at least one log rotation policy configured. The following policies are available:

- **Time Limit Rotation Policy** – Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every seven days.
- **Fixed Time Rotation Policy** – Rotates the logs every day at a specified time (based on 24-hour). The default time is 2359.
- **Size Limit Rotation Policy** – Rotates the logs when the file reaches the maximum size. The default size limit is 100 MB.
- **Never Rotate Policy** – Used in a rare event that does not require log rotation.

Configure the log rotation policy

Use `dsconfig` to modify the log rotation policy for the access logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configure the log retention policy

Use `dsconfig` to modify the log retention policy for the access logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configure log listeners

The server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed.

Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true</screen>
```

The path specified by the `copy-to-directory` property must exist, and the filesystem containing that directory must have enough space to hold all of the log files that will be written there. The server will automatically monitor free disk space on the target filesystem and will generate administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the `summarize-access-log` tool on a recently-rotated log file and writes its output to a file in a specified location.

This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the `summarize-access-log` tool, including the `file-based-access` and `operation-timing-access` logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of `.summary`. If the `output-directory` property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically be deleted. Though files are generally small in comparison to the log files themselves, make sure that there is enough space available in the specified storage directory. The server automatically monitors free disk space on the filesystem to which the summary files are written.

System alarms, alerts, and gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server available' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a `Condition` property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a `Probable Cause` property and `Alarm Type` property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

PingData servers are compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when Condition and Resource property are the same. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and Administrative Console.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

Alert handlers

Alert notifications can be sent to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The server provides a number of alert handler implementations configured with the `dsconfig` tool, including:

- **Error Log Alert Handler** – Sends administrative alerts to the configured server error logger(s).
- **JMX Alert Handler** – Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. The server uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SNMP Alert Handler** – Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.

If needed, the Server SDK can be used to implement additional, third-party alert handlers.

Configure alert handlers

Alert handlers can be configured with the `dsconfig` tool. PingData servers support JMX, SMTP, and SNMP. Use the `--help` option for a list of configuration options. The following is a sample command to create and enable an SMTP Alert handler from the command line:

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Test alerts and alarms

After alarms and alert handlers are configured, verify that the server takes the appropriate action when an alarm state changes by manually increasing the severity of a gauge. Alarms and alerts can be verified with the `status` tool.

1. Configure a gauge with `dsconfig` and set the `override-severity` property to critical. The following example uses the CPU Usage (Percent) gauge.

   ```
   $ dsconfig set-gauge-prop \
     --gauge-name "CPU Usage (Percent)" \
     --set override-severity:critical
   ```

2. Run the `status` tool to verify that an alarm was generated with corresponding alerts. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

   ```
   $ bin/status
   ```

   ```
                           --- Administrative Alerts ---
   Severity : Time           : Message
   ---------:----------------:-----------------------------------------------
   ```

```
-------
Error    : 11/Aug/2016    : Alarm [CPU Usage (Percent). Gauge CPU Usage
(Percent)
         : 15:41:00 -0500 : for Host System has
         :                : a current value of '18.583333333333332'.
         :                : The severity is currently OVERRIDDEN in the
         :                : Gauge's configuration to 'CRITICAL'.
         :                : The actual severity is: The severity is
         :                : currently 'NORMAL', having assumed this
severity
         :                : Mon Aug 11 15:41:00 CDT 2016. If CPU use is
high,
         :                : check the server's current workload and make
any
         :                : needed adjustments. Reducing the load on the
system
         :                : will lead to better response times.
         :                : Resource='Host System']
         :                : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
hours
Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
                                  --- Alarms ---
Severity : Severity   : Condition : Resource    : Details
         : Start Time :           :             :
---------:------------:-----------:-------------:-------------------------
--
Critical : 11/Aug/2016: CPU Usage : Host System : Gauge CPU Usage
(Percent) for
         : 15:41:00   : (Percent) : : Host System
         :  -0500     :           :             : has a current value of
         :            :           :             : '18.785714285714285'.
         :            :           :             : The severity is
currently
         :            :           :             : 'CRITICAL', having
assumed
         :            :           :             : this severity Mon Aug 11
         :            :           :             : 15:49:00 CDT 2016. If
CPU use
         :            :           :             : is high, check the
server's
         :            :           :             : current workload and
make any
         :            :           :             : needed adjustments.
Reducing
         :            :           :             : the load on the system
will
         :            :           :             : lead to better response
times
```

```
Shown are alarms of severity [Warning,Minor,Major,Critical
Use the --alarmSeverity option to filter this list
```

Use the status tool

PingData servers provides the `status` tool, which outputs the health of the server. The `status` tool polls the current health of the server and displays summary information about the number of operations processed in the network. The tool provides the following information:

Status Tool Sections

Status Section	Description
Server Status	Displays the server start time, operation status, number of connections (open, max, and total).
Server Details	Displays the server details including host name, administrative users, install path, server version, and Java version.
Connection Handlers	Displays the state of the connection handlers including address, port, protocol and current state.
Admin Alerts	Displays the 15 administrative alerts that were generated over the last 48-hour period. Limit the number of displayed alerts using the `--maxAlerts` option. For example, status `--maxAlerts 0` suppresses all alerts.

Synchronization-specific status

The `status` tool displays the following information for the PingDataSync Server.

PingDataSync Server Status Information

Status Section	Description
Sync Topology	Displays information about the connected Sync topology and any standby PingDataSync Server instances.
Summary for Sync Pipe	Displays the health status for each Sync Pipe configured on the topology. Status for each Sync Pipe includes the following: • Started – Indicates whether the Sync Pipe has started. • Current Ops Per Second – Lists the current throughput rate in operations per second. • Percent Busy – Lists the number of current operations currently divided by the number of worker threads. • Changes Detected – Lists the total number of changes detected. • Ops Completed Total – Lists the total number of changes detected and completed. • Num Ops In Flight – Lists the number of operations that are in flight. • Num Ops In Queue – Lists the number of operations that are on the input queue waiting to be synchronized. • Source Unretrieved Changes – Lists how many outstanding changes are still in the source change log that have not yet been retrieved by the PingDataSync Server. If this

Status Section	Description
	is greater than zero, it indicates a backlog, because the internal queue is too full to include these changes. • Failed Op Attempts – Lists the number of failed operation attempts. • Poll For Source Changes Count – Lists the number of times that the source has been polled for changes.
Operations Completed for the Sync Pipe	Displays the completed operation statistics for the sync pipe, including the following: • Success – Lists the number of changes that completed successfully. • Out Of Scope – Lists the number of changes that were included in the Sync Pipe, but were dropped because they did not match criteria in a Sync Class. • Op Type Not Synced – Lists the number of changes that completed because the operation type is not synchronized. • No Change Needed – Lists the number of changes that completed because no change was needed. • Entry Already Exists – Lists the number of changes that completed unsuccessfully because the entry already existed for a Create operation. • No Match Found – Lists the number of changes that completed unsuccessfully because no match for an operation (such as Modify) was found. • Multiple Matches Found – Lists the number of changes that completed unsuccessfully because multiple matches for a source entry were found at the destination. • Failed During Mapping – Lists the number of changes that completed unsuccessfully because there was a failure during attribute or DN mapping. • Failed At Resource – Lists the number of changes that completed unsuccessfully because they failed at the source. • Unexpected Exception – Lists the number of changes that completed unsuccessfully because there was an unexpected exception during processing. • Total – Lists the number of operations completed.
Sync Pipe Source Stats	Displays the source statistics for the external server, including the following: • Is Connected – Indicates whether the Sync Source is connected or not. • Connected Server – Indicates the hostname and port number of the connected server. • Successful Connect Attempts – Indicates the number of successful connection attempts. • Failed Connect Attempts – Indicates the number of failed connection attempts. • Forced Disconnects – Indicates the number of forced disconnects. • Root DSE Polls – Indicates the number of polling attempts of the root DSE. • Unretrieved Changes – Indicates the number of unretrieved changes.

Status Section	Description
	- Entries Fetched – Indicates the number of entries fetched from the source. - Failed To Decode Changelog Entry – Indicates the operations that failed to decode changelog entries. - Ops Excluded By Modifiers Name – Indicates the number of operations excluded by modifier’s name. - Num Backtrack Batches Retrieved – Indicates the number of backtrack batches retrieved.
Sync Pipe Destination Stats	Displays the destination statistics for the external server, including the following: - Is Connected – Indicates whether the Sync Source is connected or not. - Connected Server – Indicates the connection URL of the connected server. - Successful Connect Attempts – Indicates the number of successful connection attempts. - Failed Connect Attempts – Indicates the number of failed connection attempts. - Forced Disconnects – Indicates the number of forced disconnects. - Entries Fetched – Indicates the number of entries fetched. - Entries Created – Indicates the number of entries created. - Entries Modified – Indicates the number of entries modified. - Entries Deleted – Indicates the number of entries deleted.

Monitor the PingDataSync Server

The PingDataSync Server exposes its monitoring information under the `cn=monitor` entry. Various tools can be used to surface the server’s information including the PingDataMetrics Server, the Administrative Console, JConsole, LDAP command-line tools, or SNMP. The following information is collected for the PingDataSync Server. To configure the PingData PingDataMetrics Server to display PingDataSync Server data, see the *Ping IdentityPingDataMetrics Server Administration Guide*.

PingDataSync Server Monitoring Component

Component	Description
Active Operations	Provides information about the operations currently being processed by the server including the number of operations, information about the operation, and the number of active persistent searches.
Backend	Provides general information about the state of the server backend, including the backend ID, base DN(s), entry counts, entry count for the `cn=admin` data, writability mode, and whether it is a private backend. The following backend monitors are provided: - adminRoot

Component	Description
	• ads-truststore • alerts • backup • config • monitor • schema • tasks • userRoot
Berkeley DB JE Environment	Provides information about the state of the Oracle Berkeley DB Java Edition database used by the PingDataSync Server backend.
Client Connections	Provides information about all client connections to the server.
Disk Space Usage	Provides information about the disk space available to various components of the server. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the `DiskSpaceConsumer` interface.
Connection Handler	Provides information about the available connection handlers on the server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections.
General	Provides general information about the server, including product name and server version.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages, and operations handled.
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time, and number of processing times between `0ms` and `1ms`.
System Information	Provides general information about the system and the JVM on which the server is running, including host name, operating system, JVM architecture, Java home, and Java version.
Version	Provides information about the product version, including build ID and revision number.
Work Queue	Provides information about the state of the PingDataSync Server work queue, which holds requests until they can be processed by a worker thread. The work queue configuration has a `monitor-queue-time` property set to `true` by default. This logs messages for new operations with a `qtime` attribute included in

Component	Description
	the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations be processed using the administrative thread pool, the requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

Chapter 10: Troubleshooting

There are several ways to troubleshoot issues with the PingDataSync Server.

Topics include:

Synchronization troubleshooting

The majority of synchronization problems involve the connection state of the external servers and the synchronization of the data between the two endpoints. Make sure the PingDataSync Server can properly fail over to another endpoint or server instance if the connection fails on the highest priority external server.

Another factor in troubleshooting synchronization is determining if the DN and attribute mappings were properly configured and if the information is properly being synchronized across the network. Typical scenarios include checking for any entry failures and mapping issues.

> **Note**
> Use the `resync` tool to validate Sync Classes and data mappings from one endpoint to another. The tool provides a `dry-run` option that verifies data operations without actually affecting the data.

The following log files are specific to the PingDataSync Server, and contain details about the synchronization processes:

- **Sync Log** – provides information about the synchronization operations that occur within the server. Specifically, the Sync Log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization. The log also shows the entries that were involved in the synchronization process.
- **Sync Failed Operations Log** – provides a list of synchronization operations that have failed.
- **Resync Log** – provides summaries or details of synchronized entries and any missing entries in the Sync Destination.
- **Resync Error Log** – provides error information for `resync` operations.

Management tools

Each PingData server provides command-line tools to manage, monitor, and diagnose server operations. Each tool provides a description of the subcommands, arguments, and usage examples needed to run the tool.

> **Note**
> For detailed information and examples of the command-line tools, see the Configuration Reference Guide in the `<server-root>/docs` directory, or linked from the Administrative Console.

To view detailed argument options and examples, use `--help` with the each tool:

```
$ bin/dsconfig --help
```

For those utilities that support additional subcommands (such as `dsconfig`), list the subcommands with the following:

```
$ bin/dsconfig --help-subcommands
```

View more detailed subcommand information by using `--help` with the specific subcommand:

```
$ bin/dsconfig list-log-publishers --help
```

Troubleshooting tools

PingData provides utilities to troubleshoot the state of each server and to determine the cause of any issues. The following tools are available for diagnosing any problems and are located in the `<server-root>/bin` directory, or the `<server-root>/bat` directory on Windows systems:

Troubleshooting Tools

Tool	Description
`status`	Provides a high-level view of the current operational state of the server and displays any recent alerts that have occurred in past 24 hours.
`ldap-diff`	Used to compare one or more entries across two server endpoints to determine data issues.
`ldapsearch`	Retrieves the full entries from two different servers to determine the exact content of an entry from each server.
`logs`	The logs directory provides important logs that should be used to troubleshoot or monitor any issue with the server. Logs include server-specific operations and the following general logs: • Error Log – Provides information about warnings, errors, or significant events that occur within the server. • Debug Log – Provides detailed information, if enabled, about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database. • Access loggers – Provide information about LDAP operations processed within the server. This log only applies to operations performed in the server. This includes configuration changes, searches of monitor data, and bind operations for authenticating administrators using the command-line tools and the Administrative Console.
`collect-support-data`	Used to aggregate the results of various support tools data for the Ping IdentitySupport team to diagnose. For more information, see Using the collect-support-data Tool.
`config-diff`	Generate a summary of the configuration changes in a local or remote server instance. The tool can be used to compare configuration settings when troubleshooting issues, or when verifying configuration settings on new servers.

Use the status tool

PingData servers provides the `status` tool, which outputs the health of the server. The `status` tool polls the current health of the server and displays summary information about the number of operations processed in the network. The tool provides the following information:

Status Tool Sections

Status Section	Description
Server Status	Displays the server start time, operation status, number of connections (open, max, and total).
Server Details	Displays the server details including host name, administrative users, install path, server version, and Java version.
Connection Handlers	Displays the state of the connection handlers including address, port, protocol and current state.
Admin Alerts	Displays the 15 administrative alerts that were generated over the last 48-hour period. Limit the number of displayed alerts using the `--maxAlerts` option. For example, status `--maxAlerts 0` suppresses all alerts.

Use the collect-support-data tool

PingData servers provide information about their current state and any problems encountered. If a problem occurs, run the `collect-support-data` tool in the `/bin` directory. The tool aggregates all relevant support files into a zip file that can be sent to a support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools for the operating system.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool may vary between systems. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Perform the following steps to run this tool:

1. Navigate to the server root directory.
2. Run the `collect-support-data` tool. Include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data \
  --hostname 100.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager"
  --bindPassword secret \
  --serverRoot /opt/PingData<server> \
  --pid 1234
```

3. Email the zip file to a support provider.

Use the Sync log

The Sync log, located in the logs directory (`<server-root>/logs/sync`), provides useful troubleshooting information on the type of operation that was processed or completed. Most log entries provide the following common elements in their messages:

Sync Log Elements

Sync Log Element	Description
category	Indicates the type of operation, which will always by SYNC.
severity	Indicates the severity type of the message: INFORMATION, MILD_WARNING, SEVERE_WARNING, MILD_ERROR, SEVERE_ERROR, FATAL_ERROR, DEBUG, or NOTICE.
msgID	Specifies the unique ID number assigned to the message.
op	Specifies the operation number specific to the PingDataSync Server.
changeNumber	Specifies the change number from the source server assigned to the modification.
replicationCSN	Specifies the replication change sequence number from the source server.
replicaID	Specifies the replica ID from the source server if there are multiple backend databases.
pipe	Specifies the Sync Pipe that was used for this operation.
msg	Displays the result of this operation.

Sync log example 1

The following example displays an informational message that a modification to an entry was detected on the source server.

```
$ tail -f logs/sync

[17/May/2015:15:46:19 -0500] category=SYNC severity=INFORMATION
msgID=1893728293
op=14 changeNumber=15 replicationCSN=00000128A7E3C7D31E960000000F
replicaID=7830
pipe="DS1 to DS2" msg="Detected MODIFY of uid=user.993,ou=People,dc=example,
dc=com at ldap://server1.example.com:1389"
```

Sync log example 2

The next example shows a successful synchronization operation that resulted from a MODIFY operation on the source server and synchronized to the destination server.

```
[18/May/2015:13:54:04 -0500] category=SYNC severity=INFORMATION
msgID=1893728306 op=701
changeNumber=514663 replicationCSN=00000128ACC249A31E960007DA67 replicaID=7830
pipe="DS1 to DS2" class="DEFAULT" msg="Synchronized MODIFY of
uid=user.698,ou=People,
dc=example,dc=com at ldap://server1.example.com:1389 by modifying entry
uid=user.698,
ou=People,dc=example,dc=com at ldap://server3.example.com:3389"
```

Sync log example 3

The next example shows a failed synchronization operation on a MODIFY operation from the source server that could not be synchronized on the destination server. The log displays the LDIF-formatted modification that failed, which came from a schema violation that resulted from an incorrect attribute mapping (`telephoneNumber -> telephone`) from the source to destination server.

```
[18/May/2015:11:29:49 -0500] category=SYNC severity=SEVERE_WARNING
msgID=1893859389
op=71831 changeNumber=485590 replicationCSN=00000128AC3DE8D51E96000768D6
replicaID=7830 pipe="DS1 to DS2" class="DEFAULT" msg="Detected MODIFY of
uid=user.941,ou=People,dc=example,dc=com at ldap://server1.example.com:1389,
but
failed to apply this change because: Failed to modify entry uid=user.941,
ou=People,dc=example,dc=com on destination 'server3.example.com:3389'.
Cause: LDAPException(resultCode=65(object class violation), errorMessage='
Entry uid=user.941,ou=People,dc=example,dc=com cannot be modified because the
resulting entry would have violated the server schema: Entry
uid=user.941,ou=People,
dc=example,dc=com violates the Directory Server schema configuration because
it
includes attribute telephone which is not allowed by any of theobjectclasses
defined in that entry') (id=1893859386
ResourceOperationFailedException.java:125
Build revision=6226). Details: Source change detail:

dn: uid=user.941,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: 027167170433915
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
Equivalent destination changes:
dn: uid=user.941,ou=People,dc=example,dc=com
changetype: modify
replace: telephone
telephone: 818002279103216
Full source entry:
dn: uid=user.941,ou=People,dc=example,dc=com
objectClass: person
... (more output)
Mapped destination entry:
dn: uid=user.941,ou=People,dc=example,dc=com
telephone: 818002279103216
objectClass: person
```

```
objectClass: inetOrgPerson
... (more output) ...
```

Troubleshoot synchronization failures

While many PingDataSync Server issues are deployment-related and are directly affected by the hardware, software, and network structure used in the synchronization topology, most failures usually fall into one of the following categories:

- **Entry Already Exists** – When an add operation was attempted on the destination server, an entry with the same DN already exists.
- **No Match Found** – A match was not found at the destination based on the current Sync Classes and correlation rules (DN and attribute mapping). When this value has a high count, correlation rule problems are likely.
- **Failure at Resource** – Indicates that some other error happened during the synchronization process that does not fall into the first two categories. Typically, these errors are communication problems with a source or destination server.

Statistics for these and other types of errors are kept in the `cn=monitor` branch and can be viewed directly using the `status` command.

Troubleshoot "Entry Already Exists" failures

If there is a count for the `Entry Already Exists` statistic using the `status` tool, verify the problem in the sync log. For example, the `status` tool displays the following information:

```
           --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 1
No Match Found         : 1
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 0
Unexpected Exception   : 0
Total                  : 2
```

Verify the change by viewing the `<server-root>/logs/sync` file to see the specific operation, which could be due to someone manually adding the entry on the target server:

```
op=2 changeNumber=529277 replicationCSN=00000128AD0D9BA01E960008137D
replicaID=7830
pipe="DS1 to DS2" class="DEFAULT" msg="Detected ADD of
uid=user.1001,ou=People,
dc=example,dc=com at ldap://server1.example.com:1389, but cannot create this
```

```
entry
at the destination because an equivalent entry already exists at
ldap://server3.
example.com:3389. Details: Search using [search-criteria dn:
uid=user.1001,ou=People,
dc=example,dc=com attrsToGet: [*, dn]] returned results;
[uid=user.1001,ou=People,
dc=example,dc=com]. "
```

Perform the following steps to troubleshoot this type of problem:

1. Assuming that a possible DN mapping is ill-formed, first run the `ldap-diff` utility to compare the entries on the source and destination servers. Then look at the `ldap-diff` results with the mapping rules to determine why the original search did not find a match.

   ```
   $ bin/ldap-diff \
     --outputLDIF config-difference.ldif \
     --baseDN "dc=example,dc=com" \
     --sourceHost server1.example.com \
     --targetHost server2.example.com \
     --sourcePort 1389 \
     --targetPort 3389 \
     --sourceBindDN "cn=Directory Manager" \
     --sourceBindPassword password \
     --searchFilter "(uid=1234)"
   ```

2. Review the destination server access logs to verify the search and filters used to find the entry. Typically, the key correlation attributes are not synchronized.
3. If the mapping rule attributes are not synchronized, review the Sync Classes and mapping rules, and use the information from the `ldap-diff` results to determine why a specific attribute may not be getting updated. Some questions to answer are as follows:
   - Is there more than one Sync Class that the operation could match?
   - If using an `include-base-dn` or `include-filter` in the mapping rules, does this exclude this operation by mistake?
   - If using an attribute map, are the mappings correct? Usually, the cause of this error is in the destination mapping attribute settings. For example, if a set of correlation attributes is defined as: `dn`, `mobile`, `accountNumber`, and the `accountNumber` changes for some reason, future operations on this entry will fail. To resolve this, you either remove `accountNumber` from the rule, or add a second rule as: `dn`, `mobile`. The second rule is used only if the search using the first set of attributes fails. In this case, the entry is found and the `accountNumber` information is updated.
4. If deletes are being synchronized, check to see if there was a previous delete of this entry that was not synchronized properly. In some cases, simpler logic should be used for deletes due to the available attributes in the change logs. This scenario could cause

an entry to not be deleted for some reason, which would cause an issue when a new entry with the same DN is added later. Use this information for mapping rules to see why the original search did not find a match.

5. Look at the destination directory server access logs to verify the search and filters it used to find the entry. Typically, the key attribute mappings are not synchronized.

Troubleshoot "No Match Found" failures

If there is a count for the No Match Found statistic using the `status` tool, verify the problem in the sync log. For example, if the `status` tool displays the following:

```
              --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 1
No Match Found         : 1
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 0
Unexpected Exception   : 0
Total                  : 2
```

Verify the change in the `<server-root>/logs/sync` file to see the specific operation:

```
[12/May/2016:10:30:45 -0500] category=SYNC severity=MILD_WARNING
msgID=1893793952
op=4159648 changeNumber=6648922 replicationCSN=4beadaf4002f21150000
replicaID=8469-
ou=test,dc=example,dc=com pipe="DS1 to DS2" class="Others" msg="Detected
DELETE of
'uid=1234,ou=test,dc=example,dc=com' at ldap://server1.example.com:389, but
cannot
DELETE this entry at the destination because no matching entries were found at
ldap://
server2.example.com:389. Details: Search using [search-criteria dn:
uid=1234,ou=test,dc=alu,dc=com filter: (nsUniqueId=3a324c60-5ddb11df-80ffe681-
717b93af) attrsToGet: [*, accountNumber, dn, entryuuid, mobile, nsUniqueId,
object-
Class]] returned no results."
```

Perform the following steps to fix the issue:

1. Test the search using the filter in the error message, if displayed. For example, if the sync log specifies `filter: (nsUniqueId=3a324c60-5ddb11df-80ffe681-717b93af)`, use the `ldapsearch` tool to test the filter. If it is successful, is there anything in the

attribute mappings that would exclude this from working properly?

2. Test the search using the full DN as the base. For example, use `ldapsearch` with the full DN `(uid=1234,ou=People,dc=example,dc=com)`. If it is successful, does the entry contain the attribute used in the mapping rule?
3. If the attribute is not in the entry, determine if there is a reason why this value was not synchronized. Look at the attribute mappings and the filters used in the Sync Classes.

Troubleshoot "Failed at Resource" failures

If there is a count for the "Failed at Resource" statistic using the `status` tool, verify the problem in the sync log. For example, if the `status` tool displays the following information:

```
             --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 0
No Match Found         : 0
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 1
Unexpected Exception   : 0
Total                  : 1
```

This will register after a change is detected at the source in any of the following cases:

- If the fetch of the full source entry fails. The entry exists but there is a connection failure, server down, timeout, or something similar.
- If the fetch of the destination entry fails or if the modification to the destination fails for an exceptional reason (but not for "Entry Already Exists," "Multiple Matches Found," or "No Match Found" issues).

Verify the change by viewing the `<server-root>/logs/sync` file to see the specific operation. If any of the following result codes are listed , the server is experiencing timeout errors:

- `resultCode=timeout: errorMessage=A client-side timeout was encountered while waiting 60000ms for a search response from server server1.example.com:1389`
- `resultCode=timeout: errorMessage=An I/O error occurred while trying to read the response from the server`
- `resultCode=server down: errorMessage=An I/O error occurred while trying to read the response from the server`

- `resultCode=server down: errorMessage=The connection to server server1.example.com:1389 was closed while waiting for a response to search request SearchRequest`
- `resultCode=object class violation: errorMessage='Entry device=1234,dc=example,dc=com violates the Directory Server schema configuration because it contains undefined object class`

With these "Failure at Destination" timeout errors, look at the following settings in the Directory Server to determine if adjustments are needed:

1. For External Server Properties, check the `connect-timeout` property. This property specifies the maximum length of time to wait for a connection to be established before giving up and considering the server unavailable.
2. For the Sync Destination/Sync Source Properties, check the `response-timeout` property. This property specifies the maximum length of time that an operation should be allowed to be blocked while waiting for a response from the server. A value of zero indicates that there should be no client-side timeout. In this case, the server's default will be used.

```
$ bin/dsconfig --no-prompt --port 389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password list-external-servers \
  --property connect-timeout
```

```
External Server         : Type              : connect-timeout : response-
                                                                timeout
------------------------:-------------------:-----------------:-----------
server1.example.com:389 : sundsee-ds        : 10 s            : -
server2.example.com:389 : sundsee-ds        : 10 s            : -
server3.example.com:389 : ping-identity-ds  : 10 s            : -
server4.example.com:389 : ping-identity-ds  : 10 s            : -
```

3. For Sync Pipe Properties, check the `max-operation-attempts`, `retry-backoff-initialwait`, `retry-backoff-max-wait`, `retry-backoff-increase-by`, `retry-backoff-percentage-increase`. These Sync Pipe properties provide tuning parameters that are used in conjunction with the timeout settings. When a Sync Pipe experiences an error, it will use these settings to determine how often and quickly it will retry the operation.

```
$ bin/dsconfig --no-prompt list-sync-pipes \
  --property max-operation-attempts --property retry-backoff-initial-wait
\
  --property retry-backoff-max-wait --property retry-backoff-increase-by \
  --property retry-backoff-percentage-increase \
  --port 389 --bindDN "cn=Directory Manager" \
  --bindPassword password
```

Installation and maintenance issues

The following are common installation and maintenance issues and possible solutions.

The setup program will not run

If the `setup` tool does not run properly, some of the most common reasons include the following:

A Java Environment Is Not Available – The server requires that Java be installed on the system prior to running the `setup` tool.

If there are multiple instances of Java on the server, run the `setup` tool with an explicitly-defined value for the `JAVA_HOME` environment variable that specifies the path to the Java installation. For example:

```
$ env JAVA_HOME=/ds/java ./setup
```

Another issue may be that the value specified in the provided `JAVA_HOME` environment variable can be overridden by another environment variable. If that occurs, use the following command to override any other environment variables:

```
$ env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected Arguments Provided to the JVM – If the `setup` tool attempts to launch the java command with an invalid set of arguments, it may prevent the JVM from starting. By default, no special options are provided to the JVM when running `setup`, but this might not be the case if either the `JAVA_ARGS` or `UNBOUNDID_JAVA_ARGS` environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, run the following command:

```
$ unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Started – The `setup` tool is only intended to provide the initial configuration for the server. It will not run if it detects that it has already been run.

A previous installation should be removed before installing a new one. However, if there is nothing of value in the existing installation, the following steps can be used to run the `setup` program:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories in the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The server will not start

If the server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running – Only a single instance of the server can run at any time from the same installation root. Other administrative operations can prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The <server> could not acquire an exclusive lock on file
/ds/PingData<server>/locks/server.lock:
The exclusive lock requested for file
/ds/PingData<server>/locks/ server.lock
was not granted, which indicates that another
process already holds a shared or exclusive lock on
that file. This generally means that another instance
of this server is already running.
```

If the server is not running (and is not in the process of starting up or shutting down), and there are no other tools running that could prevent the server from being started, it is possible that a previously-held lock was not properly released. Try removing all of the files in the locks directory before attempting to start the server.

There Is Not Enough Memory Available – When the server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, the server generates an error message indicating that it could not be started.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed, the server might need to be re-configured to use a smaller amount of memory.
- Another process on the system is consuming memory and there is not enough memory to start the server. Either terminate the other process, or reconfigure the server to use a smaller amount of memory.
- The server just shut down and an attempt was made to immediately restart it. If the server is configured to use a significant amount of memory, it can take a few seconds for all of the memory to be released back to the operating system. Run the `vmstat` command and wait until the amount of free memory stops growing before restarting the server.
- If the system is configured with one or more memory-backed filesystems (such as `/tmp`), determine if any large files are consuming a significant amount of memory. If so, remove them or relocate them to a disk-based filesystem.

An Invalid Java Environment or JVM Option Was Used – If an attempt to start the server fails with 'no valid Java environment could be found,' or 'the Java environment could not be started,' and memory is not the cause, other causes may include the following:

- The Java installation that was previously used to run the server no longer exists. Update the `config/java.properties` file to reference the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation has been updated, and one or more of the options that had worked with the previous Java version no longer work. Re-configure the server to use the previous Java version, and investigate which options should be used with the new installation.
- If an `UNBOUNDID_JAVA_HOME` or `UNBOUNDID_JAVA_BIN` environment variable is set, its value may override the path to the Java installation used to run the server (defined in the `config/java.properties` file). Similarly, if an `UNBOUNDID_JAVA_ARGS` environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, explicitly unset the `UNBOUNDID_JAVA_HOME`, `UNBOUNDID_JAVA_BIN`, and `UNBOUNDID_JAVA_ARGS` environment variables before starting the server.

Any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, such as:

```
$ env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option was Used – There are a small number of arguments that can be provided when running the `bin/start-server` command. If arguments were provided and are not valid, the server displays an error message. Correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration – If a change is made to the server configuration using `dsconfig` or the Administrative Console, the server will validate the change before applying it. However, it is possible that a configuration change can appear to be valid, but does not work as expected when the server is restarted.

In most cases, the server displays (and writes to the error log) a message that explains the problem. If the message does not provide enough information to identify the problem, the `logs/config-audit.log` file provides recent configuration changes, or the `config/archived-configs` directory contains configuration changes not made through a supported configuration interface. The server can be started with the last valid configuration using the `--useLastKnownGoodConfig` option:

```
$ bin/start-server --useLastKnownGoodConfig
```

To determine the set of configuration changes made to the server since the installation, use the `config-diff` tool with the arguments `--sourceLocal --targetLocal --sourceBaseline`. The `dsconfig --offline` command can be used to make configuration changes.

Proper Permissions are Missing – The server should only be started by the user or role used to initially install the server. However, if the server was initially installed as a non-root user and then started by the root account, the server can no longer be started as a non-root user. Any new files that are created are owned by root.

If the user account used to run the server needs to change, change ownership of all files in the installation to that new user. For example, if the server should be run as the "ds" user in the "other" group, run the following command as root:

```
$ chown -R ds:other /ds/PingData<server>
```

The server has shutdown

Check the current server state by using the `bin/server-state` command. If the server was previously running but is no longer active, potential reasons may include:

- Shut down by an administrator – Unless the server was forcefully terminated, then messages are written to the error and server logs stating the reason.
- Shut down when the underlying system crashed or was rebooted – Run the `uptime` command on the underlying system to determine what was recently started or stopped.
- Process terminated by the underlying operating system – If this happens, a message is written to the system error log.
- Shut down in response to a serious problem – This can occur if the server has detected that the amount of usable disk space is critically low, or if errors have been encountered during processing that left the server without worker threads. Messages are written to the error and server logs (if disk space is available).
- JVM has crashed – If this happens, then the JVM should provide a fatal error log (a `hs_err_pid<processID>.log` file), and potentially a core file.

The server will not accept client connections

Check the current server state by using the `bin/server-state` command. If the server does not appear to be accepting connections from clients, reasons can include the following:

- The server is not running.
- The underlying system on which the server is installed is not running.
- The server is running, but is not reachable as a result of a network or firewall configuration problem. If that is the case, connection attempts should time out rather than be rejected.
- If the server is configured to allow secure communication through SSL or StartTLS, a problem with the key manager or trust manager configuration can cause connection rejections. Messages are written to the server access log for each failed connection attempt.
- The server may have reached its maximum number of allowed connections. Messages should be written to the server access log for each rejected connection attempt.

- If the server is configured to restrict access based on the address of the client, messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, it can stop listening for new requests. A message should be written to the server error log with information about the problem. Restarting the server can also solve the issue. Another option is to create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The server is unresponsive

Check the current server state by using the `bin/server-state` command. If the server process is running and appears to be accepting connections but does not respond to requests received on those connections, potential reasons for this include:

- If all worker threads are busy processing other client requests, new requests are forced to wait until a worker thread becomes available. A stack trace can be obtained using the `jstack` command to show the state of the worker threads and the waiting requests.

  If all worker threads are processing the same requests for a long time, the server sends an alert that it might be deadlocked. All threads might be tied up processing unindexed searches.
- If a request handler is busy with a client connection, other requests sent through that request handler are forced to wait until it is able to read data. If there is only one request handler, all connections are impacted. Stack traces obtained using the `jstack` command will show that a request handler thread is continuously blocked.
- If the JVM in which the server is running is not properly configured, it can spend too much time performing garbage collection. The effect on the server is similar to that of a network or firewall configuration problem. A stack trace obtained with the `pstack` utility will show that most threads are idle except the one performing garbage collection. It is also likely that a small number of CPUs is 100% busy while all other CPUs are idle. The server will also issue an alert after detecting a long JVM pause that will include details.
- If the JVM in which the server is running has hung, the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a there is a network or firewall configuration problem, communication attempts with the server will fail. A network sniffer will show that packets sent to the system are not receiving TCP acknowledgment.
- If the host system is hung or lost power with a graceful shutdown, the server will be unresponsive.

If it appears that the problem is with the server software or the JVM, work with a support provider to diagnose the problem and potential solutions.

Problems with the Administrative Console

If a problem occurs when trying to use the Administrate Console, reasons may include one of the following:

- The web application container that hosts the console is not running. If an error occurs while trying to start it, consult the logs for the web application container.
- If a problem occurs while trying to authenticate, make sure that the target server is online. If it is, the access log may provide information about the authentication failure.
- If a problem occurs while interacting with the server instance using the Administrative Console, the access and error logs for that instance may provide additional information.

Problems with SSL communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-ds` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Conditions for automatic server shutdown

All PingData servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The Directory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
  --set unrecoverable-database-error-mode:initiate-server-shutdown
```

Insufficient memory errors

If the server shuts down due to insufficient memory errors, it is possible that the allocated heap size is not enough for the amount of data being returned. Consider increasing the heap size, or reducing the number of request handler threads using the following `dsconfig` command:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTP Connection Handler" \
  --set num-request-handlers:<num-of-threads>
```

Enable JVM debugging

Enable the JVM debugging options to track garbage collection data for the system. These options can impact JVM performance, but provide valuable data to tune the server. While the `jstat` utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to provide additional detail, such as:

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

Perform the following steps to enable these options for the server:

1. On the server, navigate to the `config/java.properties` file.
2. Edit the `config/java.properties` file. Add any additional arguments to the end of the line that begins with `start-<server>.java-args`.
3. Save the file.
4. Run the following command for the new arguments to take effect the next time the server is started:

   ```
   $ bin/dsjavaproperties
   ```

Index

	PingDirectory 7.0 and PingDataGovernance 7.0
Operating Systems	
	RedHat 6.6
	RedHat 6.8
	RedHat 6.9
	RedHat 7.3
	RedHat 7.4
	CentOS 6.8
	CentOS 6.9
	CentOS 7.3
	CentOS 7.4
	SUSE Enterprise 11 SP4
	SUSE Enterprise 12 SP3
	Ubuntu 16.04 LTS
	Amazon Linux
	Windows Server 2012 R2
	Windows Server 2016
Virtual Platforms	
	VMWare vSphere & ESX 6.0
	KVM
	Amazon EC2
	Microsoft Azure (Supported by Professional Services)
	VMs must follow the same minimum system requirements as non-VM deployments. Specifically, the number of CPUs and available RAM must not be constrained. Also note that VM deployments with untuned, SAN-based file systems will not perform well.
	** The AWS Elastic File System is not supported due to its long latency times*
	** Standard support agreements do not cover performance issues related to the virtual platform.*
	** Standard support agreements DO cover performance issues that can be reproduced on a non-virtual platform*
	** Standard support agreements DO cover functionality issues on supported virtual platforms, as long as the OS and JVM/JDK versions are supported*
Java Versions	
	OpenJDK 8.x
	Oracle JDK 8.x 64-bit
App Servers	
	Apache Tomcat 7.x
	Apache Tomcat 8.x
	WildFly 9.x
Admin Browsers	
	Support for versions released in 2013 or later:
	Chrome 24+
	Microsoft Edge 20+
	Safari 4+
	Firefox 25+
	For Microsoft Internet Explorer, only version 11 is supported
End User Browsers	
	Support for versions released in 2013 or later:
	Android 4.1+
	Chrome 24+
	Microsoft Edge 20+
	Safari 4+
	Firefox 25+
	For Microsoft Internet Explorer, only version 11 is supported