Installation Guide

/ Directory Services 6.5

Latest update: 6.5.6

Mark Craig

ForgeRock AS.
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to installing ForgeRock® Directory Services software.

Table of Contents

Preface

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

The ForgeRock Common REST API works across the platform to provide common ways to access web resources and collections of resources.

This guide shows you how to install, upgrade, and remove Directory Services software components listed in "Directory Services Software Components".

Read the Release Notes before you get started.

Directory Services Software Components

Component	Description
Directory Server and Tools	Pure Java, high-performance server that can be configured as: • An LDAPv3 directory server with the additional capability to serve directory data to REST applications over HTTP. • An LDAPv3 directory proxy server providing a single point of access to underlying directory servers. • A replication server handling replication traffic with directory servers and with other replication servers, receiving, sending, and storing changes to directory data. Server distributions include command-line tools for installing, configuring, and managing servers. The tools make it possible to script all operations.
DSML Gateway	DSML support is available through the gateway, which is a Java web application that you install in a web container.
REST to LDAP Gateway	In addition to the native server support for REST over HTTP, the REST to LDAP gateway is a Java web application that lets you configure REST access to any LDAPv3 directory server.
Client Toolkit	LDAP command-line client toolkit for scripting LDAP operations. The client toolkit includes LDAP command-line tools, and command-line tools for load testing.
Java APIs	Java server-side APIs for applications that embed and server plugins that extend directory services.

Component	Description
	Java LDAP client-side APIs used internally, and for building client applications. All Java APIs are fully supported, and have Interface Stability: Evolving. In other words, when you write applications or plugins using the APIs, be prepared to adapt to incompatible changes in both major and minor releases.

For a list of available downloads, see "*Before You Install*".

Chapter 1
Before You Install

This chapter covers requirements for running Directory Services software in production. It covers the following topics:

- Downloading Directory Services software
- Choosing hardware
- Choosing an operating system
- Preparing the Java environment
- Choosing an application server when using the DSML or REST to LDAP gateway
- Assigning FQDNs when using replication
- Synchronizing System Clocks For Replication
- Using appropriately signed digital certificates

Downloading Directory Services Software

The ForgeRock BackStage download site provides access to ForgeRock releases. ForgeRock releases are thoroughly validated for ForgeRock customers who run the software in production deployments, and for those who want to try or test a given release.

"Directory Services Software" describes the available software.

Directory Services Software

File	Description
`DS-6.5.6.zip`	Cross-platform distribution of the server software. Pure Java, high-performance server that can be configured as: • An LDAPv3 directory server with the additional capability to serve directory data to REST applications over HTTP. • An LDAPv3 directory proxy server providing a single point of access to underlying directory servers.

File	Description
	- A replication server handling replication traffic with directory servers and with other replication servers, receiving and sending changes to directory data. Server distributions include command-line tools for installing, configuring, and managing servers. The tools make it possible to script all operations. By default, this file unpacks into an `opendj/` directory.
`DS-6.5.6.msi`	Microsoft Windows native installer for the server software. By default, this installs files into a `C:\Program Files (x86)\OpenDJ\` directory.
`DS_6.5.6-1_all.deb`	Server software native packages for Debian and related Linux distributions. By default, this installs files into an `/opt/opendj/` directory.
`DS-6.5.6-1.noarch.rpm`	Server software native packages for Red Hat and related Linux distributions. By default, this installs files into an `/opt/opendj/` directory.
`DS-dsml-servlet-6.5.6.war`	Cross-platform DSML gateway web archive.
`DS-rest2ldap-servlet-6.5.6.war`	Cross-platform REST to LDAP gateway web archive.
`DS-monitoring-dashboard-samples-6.5.6.zip`	Sample Grafana dashboard demonstrating how to graph DS server metrics stored in a Prometheus database. You are responsible for adapting the sample to suit your production requirements. These resources are provided for *demonstration purposes only*. Commercial support for the ForgeRock DevOps Examples is not available from ForgeRock. For details on how to try the sample dashboard, see the `README.md` file delivered inside the .zip file.

Choosing Hardware

Thanks to the underlying Java platform, Directory Services software runs well on a variety of processor architectures. Many directory service deployments meet their service-level agreements without the very latest or very fastest hardware.

Fulfilling Memory Requirements

When installing a directory server for evaluation, you need 256 MB memory (32-bit) or 1 GB memory (64-bit) available.

For installation in production, read the rest of this section. You need at least 2 GB memory for a directory server and four times the disk space needed for initial production data in LDIF format. A replicated directory server stores data, indexes for the data, operational attribute data, and historical information for replication. The server configuration trades disk space for performance and resilience, compacting and purging data for good performance and for protection against temporary outages. In addition, leave space for growth in database size as client applications modify and add entries over time.

For a more accurate estimate of the disk space needed, import a known fraction of the initial LDIF with the server configured for production. Run tests to estimate change and growth in directory data, and extrapolate from the actual space occupied in testing to estimate the disk space required in production.

Directory servers almost always benefit from caching all directory database files in system memory. Reading from and writing to memory is much faster than reading from and writing to disk storage.

For large directories with millions of user directory entries, there might not be room to install enough memory to cache everything. To improve performance in such cases, use quality solid state drives either for all directory data, or as an intermediate cache between memory and disk storage.

Fulfilling Minimum Disk Space Requirements

To evaluate DS software, make sure you have 10 GB free disk space for the software and for sample data.

The more data you have, the more disk space you need. Before deploying production systems, make sure you have enough space. For details, see "Planning for High Scale" in the *Deployment Guide*.

Choosing a Processor Architecture

Processor architectures that provide fast single thread execution tend to help Directory Services software deliver the lowest response times. For top-end performance in terms of sub-millisecond response times and of throughput ranging from tens of thousands to hundreds of thousands of operations per second, the latest x86/x64 architecture chips tend to perform better than others.

When deploying DS servers with replication enabled, allow at minimum two CPU cores per server. Allow more CPU cores per server, especially in high-volume deployments or when using CPU-intensive features such as encryption. Single CPU systems seriously limit server performance.

Chip multi-threading (CMT) processors can work well for directory servers providing pure search throughput, though response times are higher. However, CMT processors are slow to absorb hundreds or thousands of write operations per second. Their slower threads get blocked waiting on resources, and thus are not optimal for deployments with high write throughput requirements.

Fulfilling Network Requirements

On systems with fast processors and enough memory to cache directory data completely, the network can become a bottleneck. Even if a single 1 Gb Ethernet interface offers plenty of bandwidth to handle your average traffic load, it can be too small for peak traffic loads. Consider using separate interfaces for administrative traffic and for application traffic.

To estimate the network hardware required, calculate the size of the data returned to applications during peak load. For example, if you expect to have a peak load of 100,000 searches per second, each returning a full 8 KB entry, you require a network that can handle 800 MB/sec (3.2 Gb/sec) throughput, not counting other operations, such as replication traffic.

Fulfilling Storage Requirements

> **Warning**
>
> The directory server does not currently support network file systems such as NFS for database storage. Provide sufficient disk space on local storage such as internal disk or an attached disk array.

For a directory server, storage hardware must house both directory data, including historical data for replication, and server logs. On a heavily used server, you might improve performance by putting access logs on dedicated storage.

Storage must keep pace with throughput for write operations. Write throughput can arise from modify, modify DN, add, and delete operations, and from bind operations when a login timestamp is recorded, and when account lockout is configured, for example.

In a replicated topology, a directory server writes entries to disk when they are changed, and a replication server writes changelog entries. The server also records historical information to resolve potential replication conflicts.

As for network throughput, base storage throughput required on peak loads rather than average loads.

Choosing an Operating System

Directory Services 6.5 software is supported on the following operating systems:

- Linux 2.6 and later
- Microsoft Windows Server 2008, 2008 R2, 2012, 2012 R2, and 2016
- Oracle Solaris 10, 11 (SPARC, x64)

In order to avoid directory database file corruption after crashes or power failures on Linux systems, enable file system write barriers and make sure that the file system journaling mode is ordered. For details on how to enable write barriers and how to set the journaling mode for data, see the options for your file system in the **mount** command manual page.

Setting Maximum Open Files

DS servers need to be able to open many file descriptors, especially when handling thousands of client connections. Linux systems in particular often set a limit of 1024 per user, which is too low to handle many client connections to the DS server.

When setting up your DS server for production use, make sure the server can use at least 64K (65536) file descriptors. For example, when running the server as user `opendj` on a Linux system that uses `/etc/security/limits.conf` to set user level limits, you can set soft and hard limits by adding these lines to the file:

```
opendj soft nofile 65536
opendj hard nofile 131072
```

The example above assumes the system has enough file descriptors available overall. You can check the Linux system overall maximum as follows:

```
$ cat /proc/sys/fs/file-max
204252
```

Setting Maximum Inotify Watches

A directory server backend database monitors file events. On Linux systems, backend databases use the inotify API for this purpose. The kernel tunable `fs.inotify.max_user_watches` indicates the maximum number of files a user can watch with the inotify API. Make sure this tunable is set to at least 512K:

```
$ sysctl fs.inotify.max_user_watches
fs.inotify.max_user_watches = 524288
```

If this tunable is set lower than that, change it as shown in the following example:

```
$ sudo sysctl --write fs.inotify.max_user_watches=524288
[sudo] password for admin:
fs.inotify.max_user_watches = 524288
```

Preventing Interference With Antivirus Software

Prevent antivirus and intrusion detection systems from interfering with DS software.

Before using DS software with antivirus or intrusion detection software, consider the following potential problems:

Interference with normal file access

> Antivirus and intrusion detection systems that perform virus scanning, sweep scanning, or deep file inspection are not compatible with DS file access, particularly database file access.
>
> Antivirus and intrusion detection software can interfere with the normal process of opening and closing database working files. They may incorrectly mark such files as suspect to infection due to

normal database processing, which involves opening and closing files in line with the database's internal logic.

Prevent antivirus and intrusion detection systems from scanning database and changelog database files.

At minimum, configure antivirus software to whitelist the DS server database files. By default, exclude the following file system directories from virus scanning:

- `/path/to/opendj/changelogDb/` (if replication is enabled)

  Prevent the antivirus software from scanning these changelog database files.

- `/path/to/opendj/db/`

  Prevent the antivirus software from scanning database files, especially `*.jdb` files.

Port blocking

Antivirus and intrusion detection software can block ports that DS uses to provide directory services.

Make sure that your software does not block the ports that DS software uses. For details, see "Limiting System and Administrative Access" in the *Security Guide*.

Negative performance impact

Antivirus software consumes system resources, reducing resources available to other services including DS servers.

Running antivirus software can therefore have a significant negative impact on DS server performance. Make sure that you test and account for the performance impact of running antivirus software before deploying DS software on the same systems.

Preparing the Java Environment

Directory Services software consists of pure Java applications. Directory Services servers and clients run on any system with full Java support. Directory Services is tested on a variety of operating systems, and supported on those listed in "Choosing an Operating System".

Directory Services 6.5 software requires Java 8 or 11, specifically at least the Java Standard Edition runtime environment, or the corresponding Java Development Kit to compile Java plugins and applications.

Note

ForgeRock validates Directory Services software with OpenJDK and Oracle JDK, and does occasionally run sanity tests with other JDKs such as the IBM JDK and Azul's Zulu. Support for very specific Java and hardware combinations is best-effort. This means that if you encounter an issue when using a particular JVM/hardware combination, you must also demonstrate the problem on a system that is widespread and easily tested by any member of the community.

ForgeRock recommends that you keep your Java installation up-to-date with the latest security fixes.

Important

Directory server JE database backends can require additional JVM options. When running a directory server with a 64-bit JVM and less than 32 GB maximum heap size, you must use the Java option, -XX:+UseCompressedOops. To use the option, edit the config/java.properties file. The following example settings include the option with the arguments for offline LDIF import, for rebuilding backend indexes, and for starting the directory server:

```
import-ldif.offline.java-args=-server -XX:+UseCompressedOops
rebuild-index.offline.java-args=-server -XX:+UseCompressedOops
start-ds.java-args=-server -XX:+UseCompressedOops
```

Make sure you have a required Java environment installed on the system. If your default Java environment is not appropriate, set OPENDJ_JAVA_HOME to the path to the correct Java environment, or set OPENDJ_JAVA_BIN to the absolute path of the **java** command. The OPENDJ_JAVA_BIN environment variable is useful if you have both 32-bit and 64-bit versions of the Java environment installed, and want to make sure you use the 64-bit version.

Running in a Container

For some settings, DS servers depend on system information reported by the JVM to determine defaults. When running DS servers in containers such as Docker, the Java 8 JVM returns information about the operating system that does not reflect container constraints and limits. When using Java 8, manually adjust the settings described below.

Note

Java 11 supports gathering container information, as described in JDK-8146115. This fix was backported to Java 8 update 191, as mentioned in the JDK 8u191 Update Release Notes.

Skip this section when using Java 8 update 191 or later, or Java 11.

If necessary, override automatic CPU detection by specifying the number of CPUs the JVM uses with -XX:ActiveProcessorCount=*count* in config/java.properties.

Before adjusting settings, determine the following container constraints:

- The number of CPU core hardware threads dedicated to the containerized system, which is usually twice the number of CPU cores

- The amount of RAM dedicated to the containerized system

When running DS servers in containers such as Docker, adjust the following settings:

- `num-request-handlers`

  Recommendation: Set this either to 2 or to 1/4 of the number of core hardware threads, whichever is larger.

- `num-worker-threads`

  Recommendation: Set this either to 4 or to 5/8 of the number of core hardware threads, whichever is larger.

- `db-num-cleaner-threads`

  Recommendation: Set this either to 2 or to 1/4 of the number of core hardware threads, whichever is larger.

- `num-update-replay-threads`

  Recommendation: Set this either to 4 or to 1/2 of the number of core hardware threads, whichever is larger.

- `-Xmx` (Java setting limiting maximum heap size)

  To use the option, edit the `config/java.properties` file and restart the server.

  For example, consider a container limited to 8 GB RAM. The following setting limits the maximum heap size to 8 GB when starting the directory server:

  ```
  start-ds.java-args=-server -Xmx8G
  ```

Choosing an Application Server

DS servers run as standalone Java services, and do not depend on an application server.

The REST to LDAP and DSML gateway applications run on Apache Tomcat (Tomcat) and Jetty.

ForgeRock supports only stable application container releases. See the Tomcat and Jetty documentation for details about the right container to use with your Java environment.

Assigning FQDNs For Replication

Directory Services replication requires use of fully qualified domain names (FQDNs), such as `opendj.example.com`.

Host names like `my-laptop.local` are acceptable for evaluation. In production, and when using replication across systems, you must either ensure DNS is set up correctly to provide FQDNs, or update the hosts file (`/etc/hosts` or `C:\Windows\System32\drivers\etc\hosts`) to supply unique, FQDNs.

Synchronizing System Clocks For Replication

When using DS replication, keep server system clocks synchronized.

To keep the system clocks synchronized, use a tool that always moves the clock forwards. For example, `ntpd` adjusts the size of a second so that time always moves forwards to eventual clock consistency.

Never move the system clock *backwards*. Never use tools such as **ntpdate** that may move the clock backwards.

Getting Digital Certificates Signed

If you plan to configure SSL or TLS to secure network communications between the server and client applications, install a properly signed digital certificate that your client applications recognize, such as one that works with your organization's PKI or one signed by a recognized certificate authority.

To use the certificate during installation, the certificate must be located in a file-based keystore supported by the JVM (JKS, JCEKS, PKCS#12), or on a PKCS#11 token. To import a signed certificate into a keystore, use the Java **keytool** command.

For details, see "Preparing For Secure Communications" in the *Administration Guide*.

Special Requests

If you have a special request regarding support for a combination not listed here, contact ForgeRock at info@forgerock.com.

Chapter 2
Installing Server Software Files

Follow the appropriate procedure for your operating system:

- "To Unpack Server Software From the Cross-Platform Zip"
- "To Install Server Software From the Debian Package"
- "To Install Server Software From the RPM Package"
- "To Install Server Software With the Windows Installer Package"

To Unpack Server Software From the Cross-Platform Zip

You can use the .zip delivery on any supported operating system.

Installation is a multi-stage process. You unpack the server software with the **unzip** command. You set up the server with the **setup** command:

1. Prepare for installation as described in "*Before You Install*".

2. Unpack the cross-platform .zip file in the file system directory where you want to install the server.

   The **setup** command, described in "*setup — install OpenDJ server*", uses the directory where you unzipped the files as the installation directory, and does not ask you where to install the server. If you want to install elsewhere on the file system, unzip the files in that location.

   Unzipping the .zip file creates a top-level `opendj` directory in the directory where you unzipped the file. On Windows systems if you unzip the file with Right-Click > Extract All, remove the trailing `opendj-6.5.6` directory from the folder you specify.

3. Run the **setup** command to set up the server.

To Install Server Software From the Debian Package

On Debian and related Linux distributions such as Ubuntu, you can install server software from the Debian package.

Installation is a multi-stage process. You install the software using the system package manager. You set up the server with the **setup** command:

1. Prepare for installation as described in "*Before You Install*".

In particular, install a Java runtime environment (JRE) if none is installed yet.

DS software requires a supported Java environment listed in "Preparing the Java Environment". The following example uses the default JRE on a system where the default version is recent enough:

```
$ sudo apt-get install default-jre
```

2. Install the server package:

   ```
   $ sudo dpkg -i DS*.deb
   ```

   The Debian package installs server files in the `/opt/opendj` directory, generates service management scripts, adds documentation files under `/usr/share/doc/opendj`, and adds man pages under `/opt/opendj/share/man`.

   The files are owned by root by default, making it easier to have the server listen on ports such as 389 and 636.

3. Set up the server with the **setup** command, **sudo /opt/opendj/setup**.

To Install Server Software From the RPM Package

On Red Hat and related Linux distributions such as Fedora and CentOS, you can install server software from the RPM package.

Installation is a multi-stage process. You install the software using the system package manager. You set up the server with the **setup** command:

1. Prepare for installation as described in "*Before You Install*".

   In particular, install a Java runtime environment (JRE) if none is installed yet.

   DS software requires a supported Java environment listed in "Preparing the Java Environment". You might need to download an RPM to install the Java runtime environment, and then install the RPM by using the **rpm** command:

   ```
   $ su
   Password:
   # rpm -ivh jre-*.rpm
   ```

2. Install the server package:

   ```
   # rpm -i DS*.rpm
   ```

The RPM package installs server files in the `/opt/opendj` directory, generates service management scripts, and adds man pages under `/opt/opendj/share/man`.

The files are owned by root by default, making it easier to have the server listen on ports such as 389 and 636.

3. Set up the server with the **setup** command, **/opt/opendj/setup**.

   By default, the server starts in run levels 2, 3, 4, and 5.

To Install Server Software With the Windows Installer Package

You can install server software on Windows using the .msi installer package.

Installation is a multi-stage process. You install the software with the Windows installer package wizard. You set up the server with the **setup** command:

1. Prepare for installation as described in "*Before You Install*".

   Prevent antivirus and intrusion detection systems from interfering with DS software.

   Before using DS software with antivirus or intrusion detection software, consider the following potential problems:

   Interference with normal file access

   Antivirus and intrusion detection systems that perform virus scanning, sweep scanning, or deep file inspection are not compatible with DS file access, particularly database file access.

   Antivirus and intrusion detection software can interfere with the normal process of opening and closing database working files. They may incorrectly mark such files as suspect to infection due to normal database processing, which involves opening and closing files in line with the database's internal logic.

   Prevent antivirus and intrusion detection systems from scanning database and changelog database files.

   At minimum, configure antivirus software to whitelist the DS server database files. By default, exclude the following file system directories from virus scanning:

   - `/path/to/opendj/changelogDb/` (if replication is enabled)

     Prevent the antivirus software from scanning these changelog database files.

   - `/path/to/opendj/db/`

     Prevent the antivirus software from scanning database files, especially `*.jdb` files.

Port blocking

Antivirus and intrusion detection software can block ports that DS uses to provide directory services.

Make sure that your software does not block the ports that DS software uses. For details, see "Limiting System and Administrative Access" in the *Security Guide*.

Negative performance impact

Antivirus software consumes system resources, reducing resources available to other services including DS servers.

Running antivirus software can therefore have a significant negative impact on DS server performance. Make sure that you test and account for the performance impact of running antivirus software before deploying DS software on the same systems.

2. Install the server files in one of the following ways:

   - Install using the MSI package:

     a. Double-click the Windows installer package, `DS-6.5.6.msi`, to start the install wizard.

     b. In the Destination Folder screen, set the folder where the wizard installs the server files.

        The default location is under Program Files on the system drive. For example, if the system drive is C:, the default location is `C:\Program Files (x86)\opendj\`, as the native executable is a 32-bit application, though you can run the server in a 64-bit Java environment.

   - Use the Microsoft **msiexec.exe** command to install the files.

     The following example installs the server files under `C:\opendj-6.5.6`, writing an installation log file, `install.log`, in the current folder:

     ```
     C:\>msiexec /i DS-6.5.6.msi /l* install.log /q OPENDJ=C:\opendj-6.5.6
     ```

3. Start the installation.

   When installation is finished, the server files are found in the location you specified as Destination Folder. You must still run the **setup** command before you can use the server.

4. Run the **setup** command to set up the server.

Chapter 3

Installing a Directory Server

This chapter covers installation of *directory servers*. Directory servers store local copies of user data, and can be replicas of other directory servers.

Directory servers can be protected from directory client access by *directory proxy servers*. Directory proxy servers hide the implementation details of a directory server deployment from client applications. For details on installing a standalone directory proxy server, see "*Installing a Directory Proxy Server*".

Directory server replicas send updates to and receive updates from *replication servers*, which are servers that do not store user data, but instead are dedicated to transmitting replication messages. A directory server can run a local replication server in the same process. Alternatively, it can connect to a replication server running in another process, either on the same system or on a remote system. For details on installing a standalone replication server, see "*Installing a Replication Server*".

Setting Up a Directory Server

Use the "*setup — install OpenDJ server*" command-line tool. When used without subcommands or options, the command is interactive. For hints about setup choices, see "Directory Server Setup Parameters".

When performing a non-interactive, silent installation, specify at least all mandatory options as part of the command.

The following options are mandatory.

If you use only these options, the command sets up a server listening only on an administration port. The administration port is protected by a key pair generated at setup time with a self-signed certificate:

- `--adminConnectorPort {port}` (conventional port number: 4444)
- `--hostname {hostname}`
- `--rootUserDN {rootUserDN}` (default: `cn=Directory Manager`)
- `--rootUserPassword {rootUserPassword}`

To Set Up a Directory Server

After installing the server files as described in "*Installing Server Software Files*", follow these steps:

1. Run the **setup directory-server** command.

   The command is located where you installed the files, `/path/to/opendj/setup`.

   When setting up a directory server, you can optionally omit the **directory-server** subcommand.

   The following example shows non-interactive setup for evaluation. It sets a password for the default monitoring user account, `uid=Monitor`. The server listens for requests on the ports used in examples throughout the documentation. It uses a generated key pair and self-signed certificate when negotiating secure connections.

   When you set up a directory server in evaluation mode, it holds Example.com data:

```
# Set up a directory server for evaluation.
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpPort 8080 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --profile ds-evaluation \
 --acceptLicense
```

   The following example shows non-interactive setup for production. This example creates a base DN but does not import LDIF at setup time. It activates only secure traffic for HTTPS. It uses an existing key pair, rather than a generated key pair with a self-signed certificate:

```
# Set up a directory server for production.
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPasswordFile /tmp/pwd.txt \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --certNickname server-cert \
 --usePkcs12keyStore /path/to/keystore.p12 \
 --keyStorePasswordFile /tmp/keystore.pin \
 --enableStartTLS \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --baseDN dc=example,dc=com \
 --addBaseEntry \
 --productionMode \
 --acceptLicense
```

2. (Optional) Run the **status** command to review the configuration:

```
$ /path/to/opendj/bin/status --offline
```

Directory Server Setup Parameters

Parameter	Description	Subcommand or Option(s)
Type of server	A directory server holds user data. A proxy server forwards requests to remote directory servers. A replication server transmits replication messages.	**directory-server** **proxy-server** **replication-server**
Instance path	Server setup uses tools and templates installed with the software to generate the instance files required to run an instance of a server. By default, all the files are co-located. This parameter lets you separate the files. Set the instance path to place generated files in a different location from the tools, templates, and libraries you installed. Interactive setup suggests co-locating the software with the instance files. You cannot use a single software installation for multiple servers. Tools for starting and stopping the server process, for example, work with a single configured server. They do not have a mechanism to specify an alternate server location. If you want to set up another server, install another copy of the software, and run that copy's **setup** command.	`--instancePath`
Root user DN	The root user DN identifies the initial *directory superuser*. This user has privileges to perform any and all administrative operations, and is not subject to access control. It is called the root user due to the similarity to the UNIX root user. The name used in the documentation is the default name: `cn=Directory Manager`. For additional security in production environments, use a different name.	`-D, --rootUserDn`

Root user password	The root user authenticates with simple, password-based authentication. Use a strong password here unless this server is only for evaluation.	`-j, --rootUserPasswordFile`
		`-w, --rootUserPassword`

Parameter	Description	Subcommand or Option(s)
Monitor user DN	The monitor user DN identifies a user with the privilege to read monitoring data (`monitor-read`). The name used in the documentation is the default name: `uid=Monitor`.	`--monitorUserDn`
Monitor user password	The monitor user authenticates with simple, password-based authentication.	`--monitorUserPasswordFile` `--monitorUserPassword`
Harden for production use	By opting to harden the server configuration for production, you increase security. The primary cost of increased security is that evaluating the software and demonstrating features can require additional configuration. For that reason, examples in the documentation assume you do not use this option. Setting up a server in hardened production mode leads to the following settings: • The default backend database for directory servers, `userRoot`, uses data confidentiality to encrypt potentially sensitive data on disk. • Global access control allows only the following access: • Anonymous users can request the StartTLS extended operation, and the Get Symmetric Key extended operation. The Get Symmetric Key extended operation is an operation designed for DS for internal use. DS servers require Get Symmetric Key extended operation access to create and share secret keys for encryption. • Anonymous users can read the root DSE operational attributes that describe server capabilities, including among other information, what security protocols and cipher suites the server supports. • Authenticated users can read the LDAP directory schema. • Authenticated users can request the LDAP Password Modify extended operation, the Who am I? extended operation, and the Cancel extended operation. • Authenticated users can request the Pre-Read and Post-Read controls, the Subtree Delete control, and the Permissive Modify control. These controls are used by the REST to LDAP gateway.	`--productionMode`

Parameter	Description	Subcommand or Option(s)
	Authenticated users can also request the ForgeRock Transaction ID control. This is a ForgeRock-specific control for internal use that permits transmission of transaction IDs through platform components for use as a key to correlation of Common Audit events. • If the setup process creates a monitor user, this user is granted access to read monitoring data. For a longer explanation of these settings, see "Reconsider Default Global Access Control" in the *Security Guide*. • The protocol version and cipher suites for securing connections are restricted to those using strong encryption. The protocol version is restricted to `TLSv1.2`. The cipher suites used when negotiating a secure connection call for a server certificate using an elliptic curve (EC) key algorithm or an RSA key algorithm. If you provide your own keystore when setting up the server in production mode, make sure that the certificate key algorithm is EC or RSA. Otherwise the server will not be able to negotiate secure connections. For details and examples, see "To Restrict Protocols and Cipher Suites" in the *Security Guide*. • The Crypto Manager requires encrypted communication between servers. The Crypto Manager is described in "Cryptographic Key Management" in the *Security Guide*. • The anonymous HTTP authorization mechanism for REST access is disabled. As a result, REST access does not permit anonymous requests. • DS native file-based access loggers and the replication error logger have UNIX/Linux file permissions set to `600` (only the server account has read-write access to log files). This setting does not affect Common Audit loggers, such as the JSON file-based audit loggers. Adjust system settings to ensure appropriate access to files. For additional information and	

Parameter	Description	Subcommand or Option(s)
	recommendations on setting the UNIX/Linux `umask` appropriately and on setting ACLs on Windows systems, see "Setting Appropriate File Permissions" in the *Security Guide*. • The random password generator generates 10-character alphanumeric passwords. • The default password policy for normal users requires passwords at least 8 characters in length, and prevents use of common passwords. The password policy for the default directory superuser requires passwords at least 8 characters in length, prevents use of common passwords, and requires that authentication be secure to avoid exposing credentials over the network. • The CRAM-MD5 and DIGEST-MD5 SASL mechanisms are disabled.	
Fully qualified directory server host name	The server uses the fully qualified host name in self-signed certificates and for identification between replicated servers. Interactive setup suggests the hostname of the local host. If this server is only for evaluation, then you can use an FQDN such as `laptop.local`. Otherwise, use an FQDN that other hosts can resolve to reach your server, and that matches the FQDN in the server certificate.	`-h, --hostname`
Administration port	This is the service port used to configure the server and to run tasks. The port used in the documentation is 4444, which is the initial port suggested during interactive setup. If the suggested port is not free, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found.	`--adminConnectorPort`
Start the server	If you do not start the server during setup, use the **/path/to/opendj/bin/start-ds** command later.	`-O, --doNotStart`

Keystore for securing connections	Setup requires a keystore with the keys for securing connections to the administration port, and to any other secure ports you configure during setup.
	You can choose to use an existing keystore supported by the JVM, which can be either a file-based keystore
		`--useJavaKeyStore`
		`--useJceKeyStore`
		`--usePkcs11KeyStore`

Parameter	Description	Subcommand or Option(s)

	or a PKCS#11 token. The existing keystore must protect the keystore and all private keys with the same PIN or password. If you choose a PKCS#11 token, you must first configure access through the JVM, as the only input to the **setup** command is the PIN.
	If you do not have an existing keystore, the **setup** command can generate a key pair in a new PKCS#12 keystore, and self-sign the public key certificate. This is the default choice during interactive setup. Other applications will not recognize self-signed certificates unless they have explicitly trusted the certificate. For example, you import the certificate into the application's truststore, or supply a copy at runtime as a CA certificate parameter.
	Public key security is often misunderstood. Before making security choices for production systems, read "*Managing Certificates and Private Keys*" in the *Security Guide*.
		`--usePkcs12KeyStore`
		`-W, --keyStorePassword`
		`-u, --keyStorePasswordFile`

LDAP and LDAPS port	The reserved port for LDAP is 389. The reserved port for LDAPS is 636.
	Examples in the documentation use 1389 and 1636, which are accessible to non-privileged users.
	If you install the server with access to privileged ports (< 1024), and the reserved port is not yet in use, then interactive setup suggests the reserved port number. If the port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found.
	The LDAP StartTLS extended operation is a standard operation to negotiate a secure connection starting on the cleartext LDAP port.
		`-p, --ldapPort`
		`-q, --enableStartTls`
		`-Z, --ldapsPort`

HTTP and HTTPS ports	The reserved port for HTTP is 80. The reserved port for HTTPS is 443. The interactive setup initially suggests 8080 and 8443 instead. If the initially suggested port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found. Examples in the documentation use 8080 and 8443. When you enable HTTP or HTTPS at setup time, only the administrative endpoints are enabled, `/`	`--httpPort` `--httpsPort`

Parameter	Description	Subcommand or Option(s)
	admin/config, /metrics/api, and /metrics/prometheus, allowing applications to configure and monitor the server. For access to user data in a directory server, see "To Set Up REST Access to User Data" in the *Administration Guide*.	
Prepare data storage and optionally import data	One of the choices when setting up a directory server is whether to prepare for and optionally add data during setup, or to handle data storage as a separate, post-setup step. You have several options for adding directory data: • Leave the database empty if you want create backend databases and import directory data separately, after completing the setup process. For details, see "*Managing Directory Data*" in the *Administration Guide*. • Create only the base DN entry if you want to prepare a backend database, but to load directory data separately, after completing the setup process. A base DN, such as dc=example,dc=com, is the DN suffix shared by all DNs in your directory data. If the concept of base DN is new to you, briefly read "About Data In LDAP Directories" in the *Administration Guide*. Before adding directory data, you must create at least one base DN. If the directory data belongs in more than one suffix, use non-interactive mode to create multiple base DNs, or load some of the data after completing the setup process. When you choose to create a base DN entry, and therefore to create a data storage backend, interactive mode can present a choice of data storage types. If you are not sure which type to choose, briefly read "About Database Backends" in the *Administration Guide*. • Import data from an LDIF file if you already have data in LDIF and you want to load directory data as part of the setup process. LDAP data interchange format (LDIF) is the standard text format for expressing LDAP data. The documentation relies on Example.com data imported when you set up the directory server	-a, --addBaseEntry -b, --baseDn -d, --sampleData -l, --ldifFile -R, --rejectFile --skipFile

Parameter	Description	Subcommand or Option(s)
	for evaluation as shown in "To Set Up a Directory Server for Evaluation". If you have LDIF already, but the data uses attributes or object classes not defined in the default schema, choose to leave the database empty, or to create a base DN entry during setup. After setup, add schema definitions as described in "*Managing Schema*" in the *Administration Guide*, and then import the data from LDIF. When you choose to import LDIF, and therefore to create a data storage backend, interactive mode can present a choice of data storage types. If you are not sure which type to choose, briefly read "About Database Backends" in the *Administration Guide*. • Load automatically-generated sample data for testing or evaluation. This option lets you have the **setup** command generate an arbitrarily large number of similar user entries. Each user entry has a `uid` RDN like `user.`*`number`*. Each user entry's password is `password`.	
Configure the server for use with other applications	The following profiles are available by default: **AM CTS data store** Configure the directory server to hold AM Core Token Service (CTS) data For the list of supported parameters, see "To Use DS for AM Core Token Service (CTS) Data". **AM identity data store** Configure the directory server to hold AM identity data For the list of supported parameters, see "To Use DS for AM Identity Data". **IDM external repository** Configure the directory server for use as the IDM external repository For the list of supported parameters, see "To Use DS as an External IDM Repository".	`--profile` `--set`

Chapter 4
Using Directory Server Setup Profiles

This chapter explains what setup profiles allow you to achieve, and demonstrates how to use the profiles delivered with DS server software.

A *setup profile* refers to the outline form of a directory server configuration for a specific use case. Profiles greatly simplify the directory server setup process for such use cases, such as preparing a directory server to serve another ForgeRock Identity Platform™ component product.

You select a profile with the **setup directory-server --profile** option. Each profile has its own parameters, some of which have default values. You specify profile parameters with `--set` options.

Each profile option takes the form `--profile` *`name`*`[:`*`version`*`]`. Run the **setup directory-server --help** command to list the profiles and versions available. If you do not specify the optional *`:version`* portion of the argument, the **setup** command uses the current DS software version. Repeat the `--profile` option to apply multiple setup profiles.

Each profile parameter set option takes the form `--set [`*`profileName/`*`]`*`parameterName`*`:`*`value`* where:

- *`profileName/`* indicates which profile the parameter applies to.

  This name is required when you specify multiple profiles, and the parameter is available in more than one of the specified profiles.

- *`parameterName`* specifies the parameter to set.

- *`value`* specifies the value the parameter takes when the **setup** command applies the profile.

Use the **setup** command `--help-profiles` option to list available profiles. Use the `--help-profile` *`name`*`[:`*`version`*`]` option to list the parameters for the specified profile.

This chapter includes the following procedures and examples:

- "To Use DS for AM Core Token Service (CTS) Data"
- "DS for AM CTS With Token Expiration"
- "DS for AM CTS With Token Expiration (Session Capabilities Required)"
- "To Use DS for AM Configuration Data"
- "To Use DS for AM Identity Data"
- "To Use DS as an External IDM Repository"
- "To Set Up a Directory Server for Evaluation"

To Use DS for AM Core Token Service (CTS) Data

Follow these steps for each DS directory server replica before enabling replication:

1. Download DS server software as described in "Downloading Directory Services Software".

2. Install the server files as described in "*Installing Server Software Files*".

3. Run the **setup directory-server** command with the `--profile am-cts` option.

   The following example prepares a first directory server replica for AM CTS data. With this configuration, the AM reaper manages token expiration and deletion. The base DN for CTS tokens is `ou=famrecords,ou=openam-session,ou=tokens`:

   ```
   $ /path/to/opendj/setup \
    directory-server \
    --rootUserDN "cn=Directory Manager" \
    --rootUserPasswordFile /tmp/rootuser.pwd \
    --monitorUserPasswordFile /tmp/monitor.pwd \
    --hostname opendj.example.com \
    --ldapPort 1389 \
    --ldapsPort 1636 \
    --httpsPort 8443 \
    --adminConnectorPort 4444 \
    --productionMode \
    --profile am-cts \
    --set am-cts/amCtsAdminPassword:str0ngEx4mplePa55word \
    --acceptLicense
   ```

   Notice that the profile configures appropriate access rights when you use the `--productionMode` option.

   AM does not require change number indexing. Disable it after setting up replication, as described in "To Disable Change Number Indexing" in the *Administration Guide*.

   For example configurations where DS manages token expiration and deletion, see "DS for AM CTS With Token Expiration" and "DS for AM CTS With Token Expiration (Session Capabilities Required)".

   The `am-cts` profile has the following parameters:

   `amCtsAdminPassword` (required)

   Password for the administrative account that AM uses to bind to the DS server.

   The bind DN for the account is `uid=openam_cts,ou=admins,ou=famrecords,ou=openam-session,`*baseDn*.

   Default: `uid=openam_cts,ou=admins,ou=famrecords,ou=openam-session,ou=tokens`

   `backendName` (optional)

   CTS backend database name.

Default: `amCts`

`baseDn` (optional)

Base DN for CTS data.

Default: `ou=tokens`

`tokenExpirationPolicy` (optional)

How CTS manages token expiration and deletion.

Set this parameter to one of the following values:

- `am`: AM CTS reaper manages token expiration and deletion.
- `am-sessions-only`: AM CTS reaper manages SESSION token expiration and deletion. DS manages expiration and deletion for all other token types.

  AM continues to send notifications about session expiration and timeouts to agents.
- `ds`: DS manages token expiration and deletion.

  AM session-related functionality is impacted and notifications are not sent.

Default: `am`

DS for AM CTS With Token Expiration

The following example prepares a first directory server replica for AM CTS data. In this case, DS manages all token expiration and deletion:

```
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPasswordFile /tmp/rootuser.pwd \
 --monitorUserPasswordFile /tmp/monitor.pwd \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --productionMode \
 --profile am-cts \
 --set am-cts/amCtsAdminPassword:str0ngEx4mplePa55word \
 --set am-cts/tokenExpirationPolicy:ds \
 --acceptLicense
```

Notice that the profile configures appropriate access rights when you use the `--productionMode` option.

AM does not require change number indexing. Disable it after setting up replication, as described in "To Disable Change Number Indexing" in the *Administration Guide*.

DS for AM CTS With Token Expiration (Session Capabilities Required)

The following example prepares a first directory server replica for AM CTS data. In this case, the AM CTS reaper manages SESSION token expiration and deletion. DS manages expiration and deletion for all other token types:

```
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPasswordFile /tmp/rootuser.pwd \
 --monitorUserPasswordFile /tmp/monitor.pwd \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --productionMode \
 --profile am-cts \
 --set am-cts/amCtsAdminPassword:str0ngEx4mplePa55word \
 --set am-cts/tokenExpirationPolicy:am-sessions-only \
 --acceptLicense
```

Notice that the profile configures appropriate access rights when you use the `--productionMode` option.

AM does not require change number indexing. Disable it after setting up replication, as described in "To Disable Change Number Indexing" in the *Administration Guide*.

To Use DS for AM Configuration Data

Follow these steps for each DS directory server replica before enabling replication:

1. Download DS server software as described in "Downloading Directory Services Software".

2. Install the server files as described in "*Installing Server Software Files*".

3. Run the **setup directory-server** command with the `--profile am-config` option.

   The following example prepares a first directory server replica for AM configuration data. The base DN for configuration data in this example is `ou=am-config`. The bind DN for the AM configuration administrator is `uid=am-config,ou=admins,ou=am-config`:

```
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPasswordFile /tmp/rootuser.pwd \
 --monitorUserPasswordFile /tmp/monitor.pwd \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --productionMode \
 --profile am-config \
 --set am-config/amConfigAdminPassword:str0ngEx4mplePa55word \
 --acceptLicense
```

The `am-config` profile has the following parameters:

`amConfigAdminPassword` (required)

Password for the administrative account that AM uses to bind to the DS server.

The bind DN for the account is `uid=am-config,ou=admins,`*`baseDn`*.

`backendName` (optional)

Configuration store backend database name.

Default: `cfgStore`

`baseDn` (optional)

Configuration store base DN.

Default: `ou=am-config`

To Use DS for AM Identity Data

Follow these steps for each DS directory server replica before enabling replication:

1. Download DS server software as described in "Downloading Directory Services Software".

2. Install the server files as described in "*Installing Server Software Files*".

3. Run the **setup directory-server** command with the `--profile am-identity-store` option.

   The following example prepares a first directory server replica for AM identity data:

```
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPasswordFile /tmp/rootuser.pwd \
 --monitorUserPasswordFile /tmp/monitor.pwd \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --productionMode \
 --profile am-identity-store \
 --set am-identity-store/amIdentityStoreAdminPassword:str0ngEx4mplePa55word \
 --acceptLicense
```

The `am-identity-store` profile has the following parameters:

`amIdentityStoreAdminPassword` (required)

Password for the administrative account that AM uses to bind to the DS server.

The default bind DN for the account is `uid=am-identity-bind-account,ou=admins,`*`baseDn`*.

Default: `uid=am-identity-bind-account,ou=admins,ou=identities`

`backendName` (optional)

User store backend database name.

Default: `amIdentityStore`

`baseDn` (optional)

Base DN for identity data.

Default: `ou=identities`

To Use DS as an External IDM Repository

Follow these steps:

1. Download DS server software as described in "Downloading Directory Services Software".

2. Install the server files as described in "*Installing Server Software Files*".

3. Run the **setup directory-server** command with the `--profile idm-repo` option.

   The following example prepares a directory server for IDM repository data:

```
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPassword password \
 --hostname localhost \
 --ldapPort 31389 \
 --adminConnectorPort 34444 \
 --profile idm-repo \
 --set idm-repo/domain:forgerock.com \
 --acceptLicense
```

The `idm-repo` profile has the following parameters:

`backendName` (optional)

IDM repository backend database name.

Default: `idmRepo`

`domain` (optional)

Domain name translated to the base DN for IDM repository data.

When you provide a domain name, the **setup** command translates it to a domain base DN. Each domain component becomes a `dc=`*`component`* RDN of the base DN. This profile then prefixes `dc=openidm` to the result. For example, the domain `example.com` translates to the domain base DN `dc=openidm,dc=example,dc=com`.

Default: `example.com`

To Set Up a Directory Server for Evaluation

Follow these steps for each DS directory server replica before enabling replication:

1. Download DS server software as described in "Downloading Directory Services Software".

2. Install the server files as described in "*Installing Server Software Files*".

3. Run the **setup directory-server** command with the `--profile ds-evaluation` option:

```
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --enableStartTLS \
 --ldapsPort 1636 \
 --httpPort 8080 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --profile ds-evaluation \
 --acceptLicense
```

The `ds-evaluation` profile has the following parameters:

`backendName` (optional)

Example data backend database name.

Default: `dsEvaluation`

`domain` (optional)

Domain name translated to the base DN for example data.

When you provide a domain name, the **setup** command translates it to a domain base DN. Each domain component becomes a `dc=`*`component`* RDN of the base DN. For example, the default domain `example.com` translates to the domain base DN `dc=example,dc=com`.

Default: `example.com`

Chapter 5
Installing a Directory Proxy Server

This chapter covers installation of standalone *directory proxy servers*. A standalone directory proxy server forwards LDAP requests for user data to remote directory servers. Directory proxy servers make it possible to provide a single point of access to a directory service, and to hide implementation details from client applications.

Unlike standalone directory proxy servers, *directory servers* store local copies of user data, and can replicate that data with other directory servers. For details on installing a directory server, see "*Installing a Directory Server*".

Setting Up a Directory Proxy Server

Use the "*setup — install OpenDJ server*" command-line tool. When used without subcommands or options, the command is interactive. For hints about setup choices, see "Directory Proxy Server Setup Parameters".

When performing a non-interactive, silent installation, specify at least all mandatory options as part of the command.

The following options are mandatory.

If you use only these options, the command sets up a server listening only on an administration port. The administration port is protected by a key pair generated at setup time with a self-signed certificate:

- `--adminConnectorPort {port}` (conventional port number: 4444)
- `--hostname {hostname}`
- `--rootUserDN {rootUserDN}` (default: `cn=Directory Manager`)
- `--rootUserPassword {rootUserPassword}`

To Set Up a Directory Proxy Server

After installing the server files as described in "*Installing Server Software Files*", follow these steps:

1. If you have not already done so, create an account for the proxy to connect to the remote directory service.

The directory proxy server binds with this account, and then forwards LDAP requests on behalf of other users.

The proxy account must have the following on all remote directory servers:

- The same bind credentials, such as bind DN and bind password
- The right to perform proxied authorization

Examples in the documentation use the account with bind DN `cn=Proxy,ou=Apps,dc=example,dc=com` and bind password `password`:

```
dn: cn=Proxy,ou=Apps,dc=example,dc=com
cn: Proxy
objectClass: top
objectClass: applicationProcess
objectClass: simpleSecurityObject
userPassword: password
ds-privilege-name: proxied-auth
```

The following example ACI on `dc=example,dc=com` grants applications the rights to perform proxied authorization:

```
aci: (target="ldap:///dc=example,dc=com") (targetattr ="*")
  (version 3.0; acl "Allow apps proxied auth"; allow(all, proxy)
  (userdn = "ldap:///cn=*,ou=Apps,dc=example,dc=com");)
```

To grant the rights to the proxy account alone, set `userdn = ldap:///cn=Proxy,ou=Apps,dc=example,dc=com` instead.

When using DS directory services, also see "Configuring Proxied Authorization" in the *Developer's Guide* for details. Otherwise, read about how to set up proxied authorization in your directory server documentation.

2. Run the **setup proxy-server** command.

   The command is located where you installed the files, `/path/to/opendj/setup`.

   The following example sets up a directory proxy server that discovers remote servers by connecting to a replication server. It forwards all requests to public naming contexts of remote servers. (Generally this means requests targeting user data, as opposed to the proxy's configuration, schema, or monitoring statistics.) It uses the least requests load balancing algorithm:

```
$ /path/to/opendj/setup \
 proxy-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPassword password \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --adminConnectorPort 4444 \
 --replicationServer rs.example.com:4444 \
 --replicationBindDN "cn=admin,cn=Administrators,cn=admin data" \
 --replicationBindPassword password \
 --proxyUserBindDN cn=Proxy,ou=Apps,dc=example,dc=com \
 --proxyUserBindPassword password \
 --proxyUsingStartTLS \
 --useJvmTrustStore \
 --acceptLicense
```

DS proxy servers do not use ACIs for access control. Instead, they use global access control policies. By default, the access rights are configured the same as the default settings for a directory server. You no doubt need to adapt these policies for your deployment. For additional details and examples, see "Configuring Global Access Control Policies" in the *Administration Guide*.

If you are just trying out the software, you can use the `--trustAll` option. Do not use this option in production environments, however.

The following example sets up a directory proxy server that has a static list of remote servers to connect to. It forwards only requests targeting `dc=example,dc=com`. It uses the default affinity load balancing algorithm:

```
$ /path/to/opendj/setup \
 proxy-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPassword password \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --adminConnectorPort 4444 \
 --staticPrimaryServer local-data-center-ldap1.example.com:636 \
 --staticPrimaryServer local-data-center-ldap2.example.com:636 \
 --staticSecondaryServer remote-data-center-ldap1.example.com:636 \
 --staticSecondaryServer remote-data-center-ldap2.example.com:636 \
 --baseDN dc=example,dc=com \
 --proxyUserBindDN cn=Proxy,ou=Apps,dc=example,dc=com \
 --proxyUserBindPassword password \
 --proxyUsingSSL \
 --useJvmTrustStore \
 --acceptLicense
```

When you set up a directory proxy server, access control is implemented using global access control policy entries, rather than global ACIs. For more information about global access control policies, see "About Global Access Control Policies" in the *Administration Guide*.

3. If you used the `--productionMode` setup option, explicitly grant appropriate access to remote data.

   The following example grants authenticated users access to read data under `dc=example,dc=com`, and to use some LDAP controls and extended operations:

   ```
   $ dsconfig \
    create-global-access-control-policy \
    --hostname opendj.example.com \
    --port 4444 \
    --bindDN "cn=Directory Manager" \
    --bindPassword password \
    --policy-name "Authenticated access to example.com data" \
    --set authentication-required:true \
    --set request-target-dn-equal-to:"dc=example,dc=com" \
    --set request-target-dn-equal-to:"**,dc=example,dc=com" \
    --set permission:read \
    --set allowed-attribute:"*" \
    --set allowed-attribute:isMemberOf \
    --set allowed-attribute-exception:authPassword \
    --set allowed-attribute-exception:userPassword \
    --set allowed-control:Assertion \
    --set allowed-control:AuthorizationIdentity \
    --set allowed-control:Noop \
    --set allowed-control:PasswordPolicy \
    --set allowed-control:PermissiveModify \
    --set allowed-control:PostRead \
    --set allowed-control:PreRead \
    --set allowed-control:ProxiedAuthV2 \
    --set allowed-control:RealAttributesOnly \
    --set allowed-control:ServerSideSort \
    --set allowed-control:SimplePagedResults \
    --set allowed-control:TransactionId \
    --set allowed-control:VirtualAttributesOnly \
    --set allowed-control:Vlv \
    --set allowed-extended-operation:PasswordModify \
    --set allowed-extended-operation:PasswordPolicyState \
    --set allowed-extended-operation:StartTls \
    --set allowed-extended-operation:WhoAmI \
    --trustAll \
    --no-prompt
   ```

   Make sure the backend directory servers have an account for the proxy and an ACI allowing proxied authorization, as described above.

   For additional examples, see "Configuring Global Access Control Policies" in the *Administration Guide*.

4. (Optional) Run the **status** command to review the configuration:

```
$ /path/to/opendj/bin/status --offline
```

5. If the LDAP schema differ on the directory servers and the proxy server, align the LDAP schema of the proxy server with the LDAP schema of the remote directory servers.

   For more information, see "*Managing Schema*" in the *Administration Guide*.

For more information, see "*Configuring LDAP Proxy Services*" in the *Administration Guide*.

Directory Proxy Server Setup Parameters

Parameter	Description	Subcommand or Option(s)
Type of server	A directory server holds user data. A proxy server forwards requests to remote directory servers. A replication server transmits replication messages.	**directory-server** **proxy-server** **replication-server**
Instance path	Server setup uses tools and templates installed with the software to generate the instance files required to run an instance of a server. By default, all the files are co-located. This parameter lets you separate the files. Set the instance path to place generated files in a different location from the tools, templates, and libraries you installed. Interactive setup suggests co-locating the software with the instance files. You cannot use a single software installation for multiple servers. Tools for starting and stopping the server process, for example, work with a single configured server. They do not have a mechanism to specify an alternate server location. If you want to set up another server, install another copy of the software, and run that copy's **setup** command.	`--instancePath`
Root user DN	The root user DN identifies the initial *directory superuser*. This user has privileges to perform any and all administrative operations, and is not subject to access control. It is called the root user due to the similarity to the UNIX root user. The name used in the documentation is the default name: `cn=Directory Manager`.	`-D, --rootUserDn`

Parameter	Description	Subcommand or Option(s)
	For additional security in production environments, use a different name.	

Root user password	The root user authenticates with simple, password-based authentication. Use a strong password here unless this server is only for evaluation.	`-j, --rootUserPasswordFile`
		`-w, --rootUserPassword`

Monitor user DN	The monitor user DN identifies a user with the privilege to read monitoring data (`monitor-read`). The name used in the documentation is the default name: `uid=Monitor`.	`--monitorUserDn`
Monitor user password	The monitor user authenticates with simple, password-based authentication.	`--monitorUserPasswordFile` `--monitorUserPassword`
Harden for production use	By opting to harden the server configuration for production, you increase security. The primary cost of increased security is that evaluating the software and demonstrating features can require additional configuration. For that reason, examples in the documentation assume you do not use this option. Setting up a server in hardened production mode leads to the following settings: • The default backend database for directory servers, `userRoot`, uses data confidentiality to encrypt potentially sensitive data on disk. • Global access control allows only the following access: • Anonymous users can request the StartTLS extended operation, and the Get Symmetric Key extended operation. The Get Symmetric Key extended operation is an operation designed for DS for internal use. DS servers require Get Symmetric Key extended operation access to create and share secret keys for encryption. • Anonymous users can read the root DSE operational attributes that describe server capabilities, including among other information, what security protocols and cipher suites the server supports. • Authenticated users can read the LDAP directory schema. • Authenticated users can request the LDAP Password Modify extended operation, the Who am	`--productionMode`

Parameter	Description	Subcommand or Option(s)
	I? extended operation, and the Cancel extended operation. • Authenticated users can request the Pre-Read and Post-Read controls, the Subtree Delete control, and the Permissive Modify control. These controls are used by the REST to LDAP gateway. Authenticated users can also request the ForgeRock Transaction ID control. This is a ForgeRock-specific control for internal use that permits transmission of transaction IDs through platform components for use as a key to correlation of Common Audit events. • If the setup process creates a monitor user, this user is granted access to read monitoring data. For a longer explanation of these settings, see "Reconsider Default Global Access Control" in the *Security Guide*. • The protocol version and cipher suites for securing connections are restricted to those using strong encryption. The protocol version is restricted to `TLSv1.2`. The cipher suites used when negotiating a secure connection call for a server certificate using an elliptic curve (EC) key algorithm or an RSA key algorithm. If you provide your own keystore when setting up the server in production mode, make sure that the certificate key algorithm is EC or RSA. Otherwise the server will not be able to negotiate secure connections. For details and examples, see "To Restrict Protocols and Cipher Suites" in the *Security Guide*. • The Crypto Manager requires encrypted communication between servers. The Crypto Manager is described in "Cryptographic Key Management" in the *Security Guide*. • The anonymous HTTP authorization mechanism for REST access is disabled. As a result, REST access does not permit anonymous requests. • DS native file-based access loggers and the replication error logger have UNIX/Linux file	

Parameter	Description	Subcommand or Option(s)
	permissions set to 600 (only the server account has read-write access to log files). This setting does not affect Common Audit loggers, such as the JSON file-based audit loggers. Adjust system settings to ensure appropriate access to files. For additional information and recommendations on setting the UNIX/Linux umask appropriately and on setting ACLs on Windows systems, see "Setting Appropriate File Permissions" in the *Security Guide*. • The random password generator generates 10-character alphanumeric passwords. • The default password policy for normal users requires passwords at least 8 characters in length, and prevents use of common passwords. The password policy for the default directory superuser requires passwords at least 8 characters in length, prevents use of common passwords, and requires that authentication be secure to avoid exposing credentials over the network. • The CRAM-MD5 and DIGEST-MD5 SASL mechanisms are disabled.	
Fully qualified directory server host name	The server uses the fully qualified host name in self-signed certificates and for identification between replicated servers. Interactive setup suggests the hostname of the local host. If this server is only for evaluation, then you can use an FQDN such as `laptop.local`. Otherwise, use an FQDN that other hosts can resolve to reach your server, and that matches the FQDN in the server certificate.	`-h, --hostname`
Administration port	This is the service port used to configure the server and to run tasks. The port used in the documentation is 4444, which is the initial port suggested during interactive setup. If the suggested port is not free, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found.	`--adminConnectorPort`

Parameter	Description	Subcommand or Option(s)
Start the server	If you do not start the server during setup, use the **/path/to/opendj/bin/start-ds** command later.	`-O, --doNotStart`

Keystore for securing connections	Setup requires a keystore with the keys for securing connections to the administration port, and to any other secure ports you configure during setup.
	You can choose to use an existing keystore supported by the JVM, which can be either a file-based keystore or a PKCS#11 token. The existing keystore must protect the keystore and all private keys with the same PIN or password. If you choose a PKCS#11 token, you must first configure access through the JVM, as the only input to the **setup** command is the PIN.
	If you do not have an existing keystore, the **setup** command can generate a key pair in a new PKCS#12 keystore, and self-sign the public key certificate. This is the default choice during interactive setup. Other applications will not recognize self-signed certificates unless they have explicitly trusted the certificate. For example, you import the certificate into the application's truststore, or supply a copy at runtime as a CA certificate parameter.
	Public key security is often misunderstood. Before making security choices for production systems, read "*Managing Certificates and Private Keys*" in the *Security Guide*.
		`--useJavaKeyStore`
		`--useJceKeyStore`
		`--usePkcs11KeyStore`
		`--usePkcs12KeyStore`
		`-W, --keyStorePassword`
		`-u, --keyStorePasswordFile`

LDAP and LDAPS port	The reserved port for LDAP is 389. The reserved port for LDAPS is 636.
	Examples in the documentation use 1389 and 1636, which are accessible to non-privileged users.
	If you install the server with access to privileged ports (< 1024), and the reserved port is not yet in use, then interactive setup suggests the reserved port number. If the port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found.
	The LDAP StartTLS extended operation is a standard operation to negotiate a secure connection starting on the cleartext LDAP port.
		`-p, --ldapPort`
		`-q, --enableStartTls`
		`-Z, --ldapsPort`

HTTP and HTTPS ports	The reserved port for HTTP is 80. The reserved port for HTTPS is 443. The interactive setup initially suggests 8080 and 8443 instead.	`--httpPort` `--httpsPort`

Parameter	Description	Subcommand or Option(s)
	If the initially suggested port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found. Examples in the documentation use 8080 and 8443. When you enable HTTP or HTTPS at setup time, only the administrative endpoints are enabled, `/admin/config`, `/metrics/api`, and `/metrics/prometheus`, allowing applications to configure and monitor the server. For access to user data in a directory server, see "To Set Up REST Access to User Data" in the *Administration Guide*.	

Connecting to directory servers	A directory proxy server uses a *service discovery mechanism* to discover and connect to remote LDAP directory servers.
	A service discovery mechanism identifies a set of directory servers that the proxy can forward requests to.
	When preparing to configure a service discovery mechanism, choose one of these alternatives:
	Replication service discovery mechanism
	This mechanism contacts DS replication servers to discover directory servers to forward LDAP requests to. Each replication server maintains information about the replication topology that allows the proxy server to discover directory server replicas.
	This mechanism only works with replicated DS servers.
	A replication service discovery mechanism configuration includes a bind DN and password to connect to replication servers. It uses this account to read configuration data under `cn=admin data` and `cn=config`. The account must have access and privileges to read that configuration data, and it must exist with the same credentials on all replication servers.
	Static service discovery mechanism
	This mechanism maintains a static list of directory server *`host:port`* combinations. You
		`--replicationBindDn`
		`--replicationBindPassword`
		`--replicationBindPasswordFile`
		`--replicationPreferredGroupId`
		`--replicationServer`
		`--staticPrimaryServer`
		`--staticSecondaryServer`

Parameter	Description	Subcommand or Option(s)
	must enumerate the servers to forward LDAP requests to. This mechanism is designed to work with all LDAPv3 directory servers that support proxied authorization. All remote directory servers are expected to be equivalent replicas of each other. In distributed deployments, nearby remote directory servers may be set as primary and others as secondary. The proxy attempts first to forward requests to primary servers. If no primary servers are available, then the proxy forwards requests to secondary servers until the primary servers become available again. This is useful, for example, to prevent a proxy from load balancing some requests over WAN links even though directory servers on the LAN are ready to receive requests. For a replication service discovery mechanism, you identify the primary server group by its replication group ID, as described in "Replication Groups" in the *Administration Guide*. For a static service discovery mechanism, you enumerate primary and secondary servers. The connection-level security (SSL, StartTLS) options for the service discover mechanism determine how the proxy secures connections to the remote directory services. Use secure connections in production deployments to avoid sending simple bind (bind DN/ password) credentials in cleartext.	

Proxy user credentials	A directory proxy server uses a proxy DN and password to connect to remote directory servers, and proxied authorization for forwarded LDAP requests. This proxy account must exist with the same credentials on all remote directory servers, and must be able to use the standard proxied authorization control.
	For details on proxied authorization and how to configure DS servers to allow it, see "Configuring Proxied Authorization" in the *Developer's Guide*.
		`--proxyUserBindDn`
		`--proxyUserBindPassword`
		`--proxyUserBindPasswordFile`
		`--proxyUsingSsl`
		`--proxyUsingStartTls`

LDAP request forwarding	A proxy can forward LDAP requests for all public naming contexts supported by remote servers, or forward only requests targeting specified base DNs. A public naming context is a subtree of user entries held by the directory server such as `dc=example, dc=com`. Public naming contexts generally include	`--baseDn`

Parameter	Description	Subcommand or Option(s)
	user data and exclude operational suffixes such as `cn=config` and `cn=schema`. Public naming contexts are published on a directory server's root DSE. The service discovery mechanism can therefore determine them dynamically.	

Load balancing algorithm	When configuring a proxy backend, choose one of these load balancing alternatives:
	`affinity`
	This load balancing algorithm routes requests with the same target DN to the same server.
	Affinity load balancing helps applications that update and then reread the same entry in quick succession. This is not a best practice, but is often seen in client applications.
	With an add or modify request on an entry that is quickly followed by a read of the entry, the requests to replicate the update can take longer than the read request, depending on network latency. Affinity load balancing forwards the read request to the same server that processed the update, ensuring that the client application obtains the expected result.
	Affinity routing depends on the values of the proxy backend property, `partition-base-dn`. The proxy consistently routes requests for entries subordinate to these entries to the same server. The values of this property should therefore be the lowest entries in your DIT that are part of the DIT structure and not part of application data. In other words, when using affinity with two main branches, `ou=groups,dc=example,dc=com` and `ou=people,dc=example,dc=com`, set:
	`partition-base-dn:ou=groups,dc=example,dc=com`
	`partition-base-dn:ou=people,dc=example,dc=com`
	In terms of the CAP theorem, affinity load balancing provides consistency and availability, but not partition tolerance. As this algorithm lacks partition tolerance, configure it to load balance requests in environments where partitions are unlikely, such as a single data center with all directory servers on the same network.
		`--loadBalancingAlgorithm`

Parameter	Description	Subcommand or Option(s)
	`least-requests` This load balancing algorithm routes requests to the LDAP directory server with the least requests currently being serviced. Least requests load balancing helps to spread requests equitably across a pool of replicated servers. In terms of the CAP theorem, least requests load balancing provides availability and partition tolerance, but not consistency. A write request followed by a read request of the same entry can be routed to a different directory server.	

Chapter 6
Installing a Replication Server

This chapter covers installation of standalone *replication servers*. Replication servers do not serve user data, but instead are dedicated to transmitting replication messages.

Directory server replicas send updates to and receive updates from replication servers. As an alternative to having standalone replication servers, a directory server can run a local replication server in the same process. For details on installing a directory server, see "*Installing a Directory Server*".

Setting Up Standalone Servers

As described in "*Managing Data Replication*" in the *Administration Guide*, some deployments require that you separate directory servers and replication servers. A standalone directory server does not relay replication messages, but only stores data. A standalone replication server only relays replication messages, and does not store user data.

When you deploy with standalone replication servers and directory servers, first set up the replication service that your directory servers connect to. Then set up the directory servers and connect them to the replication service. Follow these procedures:

- "To Set Up the First Standalone Replication Server"
- "To Add a Standalone Replication Server"
- "To Add a Standalone Directory Server"

For hints about setup choices, see "Replication Server Setup Parameters".

To Set Up the First Standalone Replication Server

Follow these steps to set up a standalone replication server as the first server in your topology:

1. Install the server files for the standalone server, as described in "*Installing Server Software Files*".
2. Set up the server as a standalone replication server:

```
$ /path/to/opendj/setup \
 replication-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPassword password \
 --hostname rs-only.example.com \
 --adminConnectorPort 4444 \
 --replicationPort 8989 \
 --acceptLicense
```

When deploying into a production environment, secure replication traffic. Use the `--secureReplication` and appropriate keystore and truststore options.

To Add a Standalone Replication Server

Follow these steps to set up a standalone replication server that connects to an existing replication server:

1. Install the server files for the standalone server, as described in "*Installing Server Software Files*".

2. Add the server to the topology as a standalone replication server:

   ```
   $ /path/to/opendj/setup \
    replication-server \
    --rootUserDN "cn=Directory Manager" \
    --rootUserPassword password \
    --hostname rs2-only.example.com \
    --adminConnectorPort 4444 \
    --replicationPort 8989 \
    --replicationServer rs-only.example.com:4444 \
    --trustAll \
    --acceptLicense
   ```

   When deploying into a production environment, secure replication traffic. Use the `--secureReplication` and appropriate keystore and truststore options.

   > **Note**
   >
   > When you add a replication server and use the **setup** command in interactive mode, the command prompts you to trust any unrecognized certificates that the remote server presents for secure communications.
   >
   > If you have not specified a truststore and you choose to trust the certificate permanently, the **setup** command stores the certificate in a file. The file is `user.home/.opendj/keystore`, where *user.home* is the

> Java system property. *user.home* is `$HOME` on Linux and UNIX, and `%USERPROFILE%` on Windows. The keystore password is `OpenDJ`. Neither the file name nor the password can be changed.

To Add a Standalone Directory Server

Follow these steps to set up a standalone directory server that connects to an existing replication server:

1. Install the server files for the standalone server, as described in "*Installing Server Software Files*".

2. Set up the server as a directory server.

   For details, see "*Installing a Directory Server*".

3. Configure replication between the standalone directory server and an existing standalone replication server.

   Notice in the following example that the standalone directory server is not a replication server (`--noReplicationServer1`), and has no replication port. Also notice that the replication server is standalone (`--onlyReplicationServer2`):

   ```
   $ /path/to/opendj/bin/dsreplication \
    configure \
    --adminUID admin \
    --adminPassword password \
    --baseDN dc=example,dc=com \
    --host1 ds-only.example.com \
    --port1 4444 \
    --bindDN1 "cn=Directory Manager" \
    --bindPassword1 password \
    --noReplicationServer1 \
    --host2 rs-only.example.com \
    --port2 4444 \
    --bindDN2 "cn=Directory Manager" \
    --bindPassword2 password \
    --replicationPort2 8989 \
    --onlyReplicationServer2 \
    --trustAll \
    --no-prompt
   ```

4. Initialize replication.

   In order to initialize replication, you must first add at least one more directory server to the topology. Otherwise, there is nowhere to replicate the data. A standalone replication server has no user data backend.

   For details on configuring data replication, see "*Managing Data Replication*" in the *Administration Guide*.

Replication Server Setup Parameters

Parameter	Description	Subcommand or Option(s)
Type of server	A directory server holds user data. A proxy server forwards requests to remote directory servers. A replication server transmits replication messages.	**directory-server** **proxy-server** **replication-server**
Instance path	Server setup uses tools and templates installed with the software to generate the instance files required to run an instance of a server. By default, all the files are co-located. This parameter lets you separate the files. Set the instance path to place generated files in a different location from the tools, templates, and libraries you installed. Interactive setup suggests co-locating the software with the instance files. You cannot use a single software installation for multiple servers. Tools for starting and stopping the server process, for example, work with a single configured server. They do not have a mechanism to specify an alternate server location. If you want to set up another server, install another copy of the software, and run that copy's **setup** command.	`--instancePath`
Root user DN	The root user DN identifies the initial *directory superuser*. This user has privileges to perform any and all administrative operations, and is not subject to access control. It is called the root user due to the similarity to the UNIX root user. The name used in the documentation is the default name: `cn=Directory Manager`. For additional security in production environments, use a different name.	`-D, --rootUserDn`

Root user password	The root user authenticates with simple, password-based authentication. Use a strong password here unless this server is only for evaluation.	`-j, --rootUserPasswordFile`
		`-w, --rootUserPassword`

Monitor user DN	The monitor user DN identifies a user with the privilege to read monitoring data (`monitor-read`).	`--monitorUserDn`

Parameter	Description	Subcommand or Option(s)
	The name used in the documentation is the default name: `uid=Monitor`.	
Monitor user password	The monitor user authenticates with simple, password-based authentication.	`--monitorUserPasswordFile` `--monitorUserPassword`
Harden for production use	By opting to harden the server configuration for production, you increase security. The primary cost of increased security is that evaluating the software and demonstrating features can require additional configuration. For that reason, examples in the documentation assume you do not use this option. Setting up a server in hardened production mode leads to the following settings: • The default backend database for directory servers, `userRoot`, uses data confidentiality to encrypt potentially sensitive data on disk. • Global access control allows only the following access: • Anonymous users can request the StartTLS extended operation, and the Get Symmetric Key extended operation. The Get Symmetric Key extended operation is an operation designed for DS for internal use. DS servers require Get Symmetric Key extended operation access to create and share secret keys for encryption. • Anonymous users can read the root DSE operational attributes that describe server capabilities, including among other information, what security protocols and cipher suites the server supports. • Authenticated users can read the LDAP directory schema. • Authenticated users can request the LDAP Password Modify extended operation, the Who am I? extended operation, and the Cancel extended operation. • Authenticated users can request the Pre-Read and Post-Read controls, the Subtree Delete control, and the Permissive Modify control. These controls are used by the REST to LDAP gateway. Authenticated users can also request the ForgeRock Transaction ID control. This is a	`--productionMode`

Parameter	Description	Subcommand or Option(s)
	ForgeRock-specific control for internal use that permits transmission of transaction IDs through platform components for use as a key to correlation of Common Audit events. • If the setup process creates a monitor user, this user is granted access to read monitoring data. For a longer explanation of these settings, see "Reconsider Default Global Access Control" in the *Security Guide*. • The protocol version and cipher suites for securing connections are restricted to those using strong encryption. The protocol version is restricted to `TLSv1.2`. The cipher suites used when negotiating a secure connection call for a server certificate using an elliptic curve (EC) key algorithm or an RSA key algorithm. If you provide your own keystore when setting up the server in production mode, make sure that the certificate key algorithm is EC or RSA. Otherwise the server will not be able to negotiate secure connections. For details and examples, see "To Restrict Protocols and Cipher Suites" in the *Security Guide*. • The Crypto Manager requires encrypted communication between servers. The Crypto Manager is described in "Cryptographic Key Management" in the *Security Guide*. • The anonymous HTTP authorization mechanism for REST access is disabled. As a result, REST access does not permit anonymous requests. • DS native file-based access loggers and the replication error logger have UNIX/Linux file permissions set to `600` (only the server account has read-write access to log files). This setting does not affect Common Audit loggers, such as the JSON file-based audit loggers. Adjust system settings to ensure appropriate access to files. For additional information and recommendations on setting the UNIX/Linux `umask` appropriately and on setting ACLs on Windows	

Parameter	Description	Subcommand or Option(s)
	systems, see "Setting Appropriate File Permissions" in the *Security Guide*. • The random password generator generates 10-character alphanumeric passwords. • The default password policy for normal users requires passwords at least 8 characters in length, and prevents use of common passwords. The password policy for the default directory superuser requires passwords at least 8 characters in length, prevents use of common passwords, and requires that authentication be secure to avoid exposing credentials over the network. • The CRAM-MD5 and DIGEST-MD5 SASL mechanisms are disabled.	
Fully qualified directory server host name	The server uses the fully qualified host name in self-signed certificates and for identification between replicated servers. Interactive setup suggests the hostname of the local host. If this server is only for evaluation, then you can use an FQDN such as `laptop.local`. Otherwise, use an FQDN that other hosts can resolve to reach your server, and that matches the FQDN in the server certificate.	`-h, --hostname`
Administration port	This is the service port used to configure the server and to run tasks. The port used in the documentation is 4444, which is the initial port suggested during interactive setup. If the suggested port is not free, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found.	`--adminConnectorPort`
Start the server	If you do not start the server during setup, use the **/path/to/opendj/bin/start-ds** command later.	`-O, --doNotStart`

Keystore for securing connections	Setup requires a keystore with the keys for securing connections to the administration port, and to any other secure ports you configure during setup.
	You can choose to use an existing keystore supported by the JVM, which can be either a file-based keystore or a PKCS#11 token. The existing keystore must protect the keystore and all private keys with the
		`--useJavaKeyStore`
		`--useJceKeyStore`
		`--usePkcs11KeyStore`
		`--usePkcs12KeyStore`
		`-W, --keyStorePassword`

Parameter	Description	Subcommand or Option(s)
	same PIN or password. If you choose a PKCS#11 token, you must first configure access through the JVM, as the only input to the **setup** command is the PIN. If you do not have an existing keystore, the **setup** command can generate a key pair in a new PKCS#12 keystore, and self-sign the public key certificate. This is the default choice during interactive setup. Other applications will not recognize self-signed certificates unless they have explicitly trusted the certificate. For example, you import the certificate into the application's truststore, or supply a copy at runtime as a CA certificate parameter. Public key security is often misunderstood. Before making security choices for production systems, read "*Managing Certificates and Private Keys*" in the *Security Guide*.	`-u, --keyStorePasswordFile`
HTTP and HTTPS ports	The reserved port for HTTP is 80. The reserved port for HTTPS is 443. The interactive setup initially suggests 8080 and 8443 instead. If the initially suggested port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found. Examples in the documentation use 8080 and 8443. When you enable HTTP or HTTPS at setup time, only the administrative endpoints are enabled, `/admin/config`, `/metrics/api`, and `/metrics/prometheus`, allowing applications to configure and monitor the server. For access to user data in a directory server, see "To Set Up REST Access to User Data" in the *Administration Guide*.	`--httpPort` `--httpsPort`

Configure replication	You configure a port where this server listens for replication messages.
	For production servers, secure replication communications. When you choose to secure replication communications during setup, the replication server negotiates TLS with the certificate and private key from the keystore that you selected during setup.
	If another replication server already exists, specify the remote server's hostname and administration
		`-b, --baseDn`
		`--replicationPort`
		`--replicationServer`
		`--secureReplication`
		`-T, --trustStorePassword`
		`-U, --trustStorePasswordFile`

Parameter	Description	Subcommand or Option(s)

	port number. This replication server must be able to connect to the existing replication server during setup. This step requires that you let the server start during setup. Notice that the remote server port is the administration port, not the replication port.
	When connecting to a remote replication server, this server uses the global administrator account for the topology. The global administrator account must have ID `admin`, and must use the same password as the root user password for this server. If you have enabled secure replication communications on the remote server, this server must be able to trust the remote server's certificate. If necessary, make sure that this server's truststore enables it to trust the remote server's certificate. (You can manually trust the remote server certificate when setting up this server in interactive mode.)
	You can restrict the user data domains that this server replicates. You do so by specifying the base DNs of each replication domain. If you do not specify any base DNs, this server replicates user data from all available domains. In addition to user data, this server also replicates administrative data: LDAP schema (`cn=schema`) and replication configuration (`cn=admin data`). You do not need to specify base DNs for LDAP schema and administrative data.
		`--useJavaTrustStore`
		`--useJceTrustStore`
		`--useJvmTrustStore`
		`--usePkcs12TrustStore`
		`-X, --trustAll`

Chapter 7
Installing the REST to LDAP Gateway

This chapter explains how to install the REST to LDAP gateway web application.

To Install the REST to LDAP Gateway

The REST to LDAP gateway functions as a web application in a web application container. The REST to LDAP gateway runs independently of the LDAPv3 directory service. As an alternative to the gateway, you can configure HTTP access to a directory server as described in "To Set Up REST Access to User Data" in the *Administration Guide*.

You configure the gateway to access your directory service by editing configuration files in the deployed web application:

`WEB-INF/classes/config.json`

This file defines how the gateway connects to LDAP directory servers, and how user identities extracted from HTTP requests map to LDAP user identities.

For details, see "Gateway Configuration File" in the *Reference*.

`WEB-INF/classes/logging.properties`

This file defines logging properties, and can be used when the gateway runs in Apache Tomcat.

`WEB-INF/classes/rest2ldap/rest2ldap.json`

This file defines which LDAP features the gateway uses.

For details, see "Gateway REST2LDAP Configuration File" in the *Reference*.

`WEB-INF/classes/rest2ldap/endpoints/api/example-v1.json`

This file defines JSON resource to LDAP entry mappings.

You can edit this file, and define additional files for alternative APIs and versions of APIs. For details, see "Mapping Configuration File" in the *Reference*.

Follow these steps to install the REST to LDAP gateway:

1. Prepare for installation as described in "*Before You Install*".

2. Deploy the .war file according to the instructions for your application server.

3. Edit the configuration files in the deployed gateway web application.

   At minimum adjust the following configuration settings in `WEB-INF/classes/config.json`:

   - `primaryLDAPServers`: Set the correct directory server host names and port numbers.
   - `authentication`: Set the correct simple bind credentials.

     The LDAP account used to authenticate needs to perform proxied authorization as described in "Configuring Proxied Authorization" in the *Developer's Guide*.

   The default sample configuration works with generated example data, and with the sample data imported when you set up the directory server for evaluation as shown in "To Set Up a Directory Server for Evaluation". If your data is different, then you must also change the JSON resource to LDAP entry mapping settings, described in "Mapping Configuration File" in the *Reference*.

   For details regarding the configuration, see "*REST to LDAP Configuration*" in the *Reference*.

   When connecting to a directory service over LDAPS or LDAP and StartTLS, you can configure the trust manager to use a file-based truststore for server certificates that the gateway should trust. This allows the gateway to validate server certificates signed, for example, by a certificate authority that is not recognized by the Java environment when setting up LDAPS or StartTLS connections. See "Preparing For Secure Communications" in the *Administration Guide* for an example of how to use the Java **keytool** command to import a server certificate into a truststore file.

4. (Optional) If necessary, adjust the log level.

   Log levels are defined in `java.util.logging.Level` .

   By default, the log level is set to `INFO`, and the gateway logs HTTP request-related messages. To have the gateway log LDAP request-related messages, set the log level to `FINEST` in one of the following ways:

   - If the REST to LDAP gateway runs in Apache Tomcat, edit `WEB-INF/classes/logging.properties` to set `org.forgerock.opendj.rest2ldap.level = FINEST`. For details on Tomcat's implementation of the logging API, see *Logging in Tomcat*.

     Messages are written to `*CATALINA_BASE*/logs/rest2ldap.*yyyy-MM-dd*.log`.

   - If the REST to LDAP gateway runs in Jetty, make sure you set the log level system property when starting Jetty: `-Dorg.forgerock.opendj.rest2ldap.level=FINEST`.

     Messages are written to the Jetty log.

5. Restart the REST to LDAP gateway or the application server to make sure the configuration changes are taken into account.

6. Make sure that the directory service is up, and then check that the gateway is connecting correctly.

The following command reads Babs Jensen's entry through the gateway to a directory server set up for evaluation as shown in "To Set Up a Directory Server for Evaluation". In this example, the gateway is deployed under `/rest2ldap`:

```
$ curl \
 --user bjensen:hifalutin \
 http://opendj.example.com:8080/rest2ldap/api/users/bjensen?_prettyPrint=true
{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : { },
  "userName" : "bjensen@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
    "givenName" : "Barbara",
    "familyName" : "Jensen"
  },
  "description" : "Original description",
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1862",
    "emailAddress" : "bjensen@example.com"
  },
  "uidNumber" : "1076",
  "gidNumber" : "1000",
  "homeDirectory" : "/home/bjensen",
  "manager" : {
    "_id" : "trigden",
    "displayName" : "Torrey Rigden"
  }
}
```

If you generated example data, Babs Jensen's entry is not included. Instead, try a generated user such as `http://user.0:password@opendj.example.com:8080/rest2ldap/api/users/user.0`.

Chapter 8

Installing the DSML Gateway

This chapter explains how to install the DSML gateway web application.

To Install the DSML gateway

The DSML gateway functions as a web application in a web application container. The DSML gateway runs independently of the directory service. You configure the gateway to access a directory service by editing parameters in the gateway configuration file, `WEB-INF/web.xml`:

1. Prepare for installation as described in "*Before You Install*".
2. Deploy the .war file according to the instructions for your application server.
3. Edit `WEB-INF/web.xml` to ensure the parameters are correct.

   At minimum, make sure the correct values are set for `ldap.host` and `ldap.port`.
4. Restart the web application according to the instructions for your application server.

Chapter 9

Before You Upgrade

This chapter lists requirements to fulfill before upgrading Directory Services server software, especially before upgrading the software in a production environment, in addition to requirements listed in "*Before You Install*". It covers the following topics:

- Supported upgrade paths
- Required credentials
- Java upgrades
- New tuning for upgraded servers
- Backup files
- Debug logging that you must disable
- Servers that run as Windows services

Following a Supported Upgrade Path

"Server Upgrade Paths" indicates what you can upgrade.

Server Upgrade Paths

From...	To...	Important Notes
Official ForgeRock release, version 2.4 or 2.5	Official ForgeRock release, directory server or replication server	Not supported. Workaround: First upgrade all servers in the deployment to use at least 2.6.0 before upgrading further. For details on upgrading to that version, see *Upgrading to OpenDJ 2.6.0*.
Official ForgeRock release, version 2.6.0 or later	Official ForgeRock release, same edition of directory server or replication server	Supported.
Official ForgeRock release, OEM edition, version 3.0.0 or later	Official ForgeRock release, directory server or replication server	Supported.

From...	To...	Important Notes
		The OEM edition did not include Berkeley DB Java Edition, and did not support JE backends. Instead, the OEM edition uses PDB backends for local data. This release removes support for PDB backend databases. The upgrade process only converts PDB backend configuration entries to JE backend configuration entries. It renames the PDB backend database directories, appending a `.bak` suffix, but does not change the format of the databases. *The PDB backend database content is no longer accessible after upgrade.* Backup archives of PDB backend databases are also no longer usable after upgrade. You must export data from any PDB backend databases to LDIF before upgrading, and then import the data into the new JE backend databases after upgrade. For instructions on exporting and importing LDIF, see "Importing and Exporting Data" in the *Administration Guide*. After upgrading, configure backup tasks for the new JE backend databases as you had done previously for PDB backend databases.
Trial edition release, version 3.0.0 or later	Official ForgeRock release, directory server or replication server	Supported.
Unofficial build, version 2.6.0 or later	Official ForgeRock release, directory server	Not supported. Workaround: Install the new directory server as a replica of other servers. Use replication to bring the new server up to date before retiring older servers.

Obtaining the Required Credentials

Perform the upgrade procedure as the user who owns the server files.

Make sure you have the credentials to run commands as this user.

Upgrading Java

Directory Services 6.5 software requires Java 8 or 11, specifically at least the Java Standard Edition runtime environment, or the corresponding Java Development Kit to compile Java plugins and applications.

If the server uses an older version, install a newer Java version before upgrading. To enable the server to use the newer Java version, edit the `default.java-home` setting in the `opendj/config/java.properties` file.

Upgrading System and Server Tuning

Major software releases include significant changes that can render existing tuning settings obsolete. When upgrading to a new major release of DS or Java software, revisit the system configuration, server configuration, and Java settings. Adjust the settings appropriately for your deployment as part of the upgrade process.

For information and suggestions on tuning, read the Release Notes and "*Tuning Servers For Performance*" in the *Administration Guide*.

Backing Up Server Files

Before upgrading, perform a full file system backup of the current server in order to revert if the upgrade fails.

Due to changes to the backup archive format, make sure you stop the directory server and *back up the file system directory where the current server is installed* rather than creating a backup archive with the **backup** command.

Disabling Debug Logging

Before upgrading a server from OpenDJ 2.6, remove all debug log targets and disable debug logging. Debug log configuration entries in version 2.6 are incompatible with later versions, and can prevent the server from starting after upgrade.

To list currently configured debug targets, use the **dsconfig list-debug-targets** command.

To remove a debug log target, use the **dsconfig delete-debug-target** command.

To disable debug logging, set the debug logger property `enabled:false` as in the following example that disables the default debug logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --set enabled:false \
 --trustAll \
 --no-prompt
```

Disabling the Server as a Windows Service

If you are upgrading the server on Windows, and it is registered as a Windows service, disable the server as a Windows service before upgrade, as in the following example:

```
C:\path\to\opendj\bat> windows-service.bat --disableService
```

After upgrade, you can enable the server as a Windows service again.

Chapter 10
Upgrading a Directory Server

This chapter shows how to upgrade a directory server. It includes the following procedures:

- "To Upgrade a Directory Server"
- "To Upgrade Replicated Servers"
- "To Add a New Replica to an Existing Topology"

> **Important**
>
> Failure to follow the upgrade instructions can result in the loss of all user data.
>
> Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.

To Upgrade a Directory Server

Follow these steps to upgrade a directory server:

1. Prepare for upgrade as described in "*Before You Upgrade*".
2. Stop the server.
3. Proceed to upgrade the server:
   - When upgrading a server installed from the cross-platform .zip:
     a. Unpack the new files over the old files as described in "*Installing Server Software Files*".
     b. Run the **upgrade** command, described in "*upgrade — upgrade OpenDJ configuration and application data*", to bring the server configuration and, if possible, user data up to date with the new software delivery.

        By default, the **upgrade** command runs interactively, requesting confirmation before making important configuration changes. For some potentially long-duration tasks, such as rebuilding indexes, the default choice is to defer the tasks until after upgrade.

        You can use the `--no-prompt` option to run the command non-interactively. In this case, the `--acceptLicense` option lets you accept the license terms non-interactively.

        When using the `--no-prompt` option, if the **upgrade** command cannot complete because it requires confirmation for a potentially long or critical task, then it exits with an error and a message about how to finish making the changes. You can add the `--force` option to force a non-interactive upgrade to continue in this case, also performing long running and critical tasks.

- When upgrading a server installed from native packages, use the system package management tools.

4. (Optional) When the mutable data mounted at runtime differs from that of the instance where you first run the **upgrade** command, upgrade only mutable data by running the command again with the `--dataOnly` option at runtime.

   The `--dataOnly` option can be useful when running the server in a Docker container, for example.

   This improvement is available when upgrading from DS 6.0.0 or later releases.

5. Start the upgraded server.

   At this point the upgrade process is complete. See the resulting `upgrade.log` file for a full list of operations performed.

   Replication updates the upgraded server with changes that occurred during the upgrade process.

   When you upgrade from version 3.0 or earlier, the upgrade process leaves the HTTP connection handler disabled.

   The newer configuration is not compatible with the previous configuration. You must rewrite your configuration according to "*REST to LDAP Configuration*" in the *Reference*, and then configure the server to use the new configuration. For details, see "RESTful Client Access Over HTTP" in the *Administration Guide*.

6. (Optional) If you disabled the server as a Windows service in order to upgrade, enable the server as a Windows service again as in the following example:

   ```
   C:\path\to\opendj\bat> windows-service.bat --enableService
   ```

To Upgrade Replicated Servers

> **Important**
>
> The directory server upgrade process is designed to support a rolling (sequential) upgrade of replicated servers.
>
> Do not upgrade all replicated servers at once in parallel, as this removes all replication changelog data simultaneously, breaking replication.

For each server in the replication topology, follow these steps:

1. Direct client application traffic away from the server to upgrade.

2. Upgrade the server.

3. After upgrading DS 5.5 and earlier, grant the global administrator account the following privileges:

   `bypass-lockdown`
   `monitor-read`
   `server-lockdown`

   The following example grants the privileges to the default global administrator account, which has DN `cn=admin,cn=Administrators,cn=admin data`:

   ```
   $ ldapmodify \
    --port 1389 \
    --hostname opendj.example.com \
    --bindDN "cn=admin,cn=Administrators,cn=admin data" \
    --bindPassword password
   dn: cn=admin,cn=Administrators,cn=admin data
   changetype: modify
   add: ds-privilege-name
   ds-privilege-name: bypass-lockdown
   ds-privilege-name: monitor-read
   ds-privilege-name: server-lockdown
   -
   ```

4. Direct client application traffic back to the upgraded server.

To Add a New Replica to an Existing Topology

Newer directory servers have updates to LDAP schemas that enable support for new features. The newer schemas are not all compatible with older servers.

When adding a new server to a replication topology with older servers and following the instructions in "Configuring Replication Settings" in the *Administration Guide*, also follow these recommendations:

1. Configure replication using the **dsreplication** command:

   - When adding a new server to a replication topology with 2.6.x servers, use the **dsreplication** command installed with a 2.6 server.

   - When adding a new server to a replication topology with 3.x and later servers, use the **dsreplication** command installed with a new server.

2. Use the `--noSchemaReplication` or the `--useSecondServerAsSchemaSource` option to avoid copying the newer schema to the older server.

   It is acceptable to copy the older schema to the newer server, though it prevents use of new features that depend on newer schema.

3. If applications depend on Internet-Draft change numbers, see "To Align Draft Change Numbers" in the *Administration Guide*.

Chapter 11
Upgrading a Directory Proxy Server

This chapter shows how to upgrade a directory proxy server.

To Upgrade a Directory Proxy Server

> **Note**
>
> Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.

Follow these steps:

1. Prepare for upgrade as described in "*Before You Upgrade*".

2. Stop the server.

3. Unpack the new files over the old files as described in "*Installing Server Software Files*".

4. Run the **upgrade** command, described in "*upgrade — upgrade OpenDJ configuration and application data*", to bring the server configuration data up to date with the new software delivery.

   By default, the **upgrade** command runs interactively, requesting confirmation before making important configuration changes.

   You can use the `--no-prompt` option to run the command non-interactively. In this case, the `--acceptLicense` option lets you accept the license terms non-interactively.

   When using the `--no-prompt` option, if the **upgrade** command cannot complete because it requires confirmation for a potentially long or critical task, then it exits with an error and a message about how to finish making the changes. You can add the `--force` option to force a non-interactive upgrade to continue in this case, also performing long running and critical tasks.

5. Start the upgraded server.

   At this point the upgrade process is complete. See the resulting `upgrade.log` file for a full list of operations performed.

6. (Optional) If you disabled the server as a Windows service in order to upgrade, enable the server as a Windows service again as in the following example:

```
C:\path\to\opendj\bat> windows-service.bat --enableService
```

Chapter 12
Upgrading a Replication Server

This chapter shows how to upgrade a standalone replication server, meaning a replication server that has no user data backends.

To Upgrade a Standalone Replication Server

> **Note**
>
> Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.

If the server holds user data, consider it a directory server and see "*Upgrading a Directory Server*" instead.

1. Prepare for upgrade as described in "*Before You Upgrade*".
2. Stop the server.
3. Unpack the new files over the old files as described in "*Installing Server Software Files*".
4. Run the **upgrade** command, described in "*upgrade — upgrade OpenDJ configuration and application data*", to bring the server configuration data up to date with the new software delivery.

   By default, the **upgrade** command runs interactively, requesting confirmation before making important configuration changes.

   You can use the `--no-prompt` option to run the command non-interactively. In this case, the `--acceptLicense` option lets you accept the license terms non-interactively.

   When using the `--no-prompt` option, if the **upgrade** command cannot complete because it requires confirmation for a potentially long or critical task, then it exits with an error and a message about how to finish making the changes. You can add the `--force` option to force a non-interactive upgrade to continue in this case, also performing long running and critical tasks.
5. Start the upgraded server.

   At this point the upgrade process is complete. See the resulting `upgrade.log` file for a full list of operations performed.
6. (Optional) If you disabled the server as a Windows service in order to upgrade, enable the server as a Windows service again as in the following example:

```
C:\path\to\opendj\bat> windows-service.bat --enableService
```

Chapter 13
Upgrading the REST to LDAP Gateway

Replace the REST to LDAP gateway with the newer version, as for a fresh installation, and rewrite the configuration to work with the new version.

For details, see "*Installing the REST to LDAP Gateway*".

Chapter 14
Upgrading the DSML Gateway

Replace the DSML gateway with the newer version, as for a fresh installation.

For details, see "*Installing the DSML Gateway*".

Chapter 15
Removing Server Software

This chapter covers uninstallation and includes the following procedures:

- "To Uninstall Cross-Platform Server Software"
- "To Uninstall the Debian Package"
- "To Uninstall the RPM Package"
- "To Uninstall the Windows Installer Package"

To Uninstall Cross-Platform Server Software

Follow these steps to remove software installed from the cross-platform .zip:

1. Log in as the user who installed and runs the server.
2. Stop replication as described in "To Stop Replication Permanently For a Replica" in the *Administration Guide*.
3. Stop the server.

   ```
   $ /path/to/opendj/bin/stop-ds --quiet
   ```

4. Delete the files manually:

   ```
   $ rm -rf /path/to/opendj
   ```

To Uninstall the Debian Package

When you uninstall the Debian package from the command-line, the server is stopped if it is running:

1. Stop replication as described in "To Stop Replication Permanently For a Replica" in the *Administration Guide*.
2. Purge the package from your system:

   ```
   $ sudo dpkg --purge opendj
   ```

3. (Optional) Remove any remaining server configuration files and directory data:

   ```
   $ sudo rm -rf /opt/opendj
   ```

To Uninstall the RPM Package

When you uninstall the RPM package from the command-line, the server is stopped if it is running:

1. Stop replication as described in "To Stop Replication Permanently For a Replica" in the *Administration Guide*.

2. Remove the package from your system:

   ```
   # rpm -e opendj
   ```

3. (Optional) Remove the server configuration files and any directory data:

   ```
   $ sudo rm -rf /opt/opendj
   ```

To Uninstall the Windows Installer Package

When you uninstall the files installed from the Windows installer package, only the installed files are removed:

1. Stop replication as described in "To Stop Replication Permanently For a Replica" in the *Administration Guide*.

2. Remove installed files in one of the following ways:

   - Use Windows Control Panel.

     a. Open Windows Control Panel and browse to the page to uninstall a program.

     b. Find the ForgeRock directory service in the list and uninstall it.

   - Use the **msiexec** command.

     The following command quietly removes installed files:

     ```
     C:\>msiexec /x DS-6.5.6.msi /q
     ```

3. (Optional) Manually remove the server configuration files and any directory data.

Part 16

Installation Reference

Find the installation tools in the directory where you unpacked the server files, such as `/path/to/opendj`.

Chapter 16.1

setup — install OpenDJ server

Synopsis

setup {subcommand} {options}

Description

This utility can be used to install an OpenDJ instance either as a directory server, a replication server or a proxy server.

Options

The setup command takes the following options:

Command options:

`--acceptLicense`

Automatically accepts the product license (if present).

Default: false

`--adminConnectorPort {port}`

Port on which the Administration Connector should listen for communication.

`-D | --rootUserDn {rootUserDN}`

DN for the initial root user for the Directory Server.

Default: cn=Directory Manager

`--instancePath {path}`

Path were the instance should be set up.

Default: /tmp

-j | --rootUserPasswordFile {rootUserPasswordFile}

Path to a file containing the password for the initial root user for the Directory Server.

--monitorUserDn {monitorUserDn}

DN of the default user allowed to query monitoring information.

Default: uid=Monitor

--monitorUserPassword {monitorUserPassword}

Password of the default user allowed to query monitoring information.

--monitorUserPasswordFile {monitorUserPasswordFile}

Path to a file containing the password for the default user allowed to query monitoring information.

-N | --certNickname {nickname}

Nickname of a keystore entry containing a certificate that the server should use when negotiating secure connections using StartTLS or SSL. Multiple keystore entries may be provided by using this option multiple times.

-O | --doNotStart

Do not start the server when the configuration is completed.

Default: false

--productionMode

Harden default configuration for production use.

Default: false

-Q | --quiet

Use quiet mode.

Default: false

-S | --skipPortCheck

Skip the check to determine whether the specified ports are usable.

Default: false

-u | --keyStorePasswordFile {keyStorePasswordFile}

Path to a file containing the keystore password. The keystore password is required when you specify an existing file-based keystore (JKS, JCEKS, PKCS#12).

--useJavaKeyStore {keyStorePath}

Path of a JKS keystore containing the certificate(s) that the server should use when negotiating secure connections using StartTLS or SSL.

--useJceKeyStore {keyStorePath}

Path of a JCEKS keystore containing the certificate(s) that the server should use when negotiating secure connections using StartTLS or SSL.

--usePkcs11KeyStore

Use certificate(s) in a PKCS#11 token that the server should use when accepting SSL-based connections or performing StartTLS negotiation.

Default: false

--usePkcs12KeyStore {keyStorePath}

Path of a PKCS#12 keystore containing the certificate(s) that the server should use when negotiating secure connections using StartTLS or SSL.

-w | --rootUserPassword {rootUserPassword}

Password for the initial root user for the Directory Server.

-W | --keyStorePassword {keyStorePassword}

Keystore cleartext password.

General options:

-V | --version

Display Directory Server version information.

Default: false

-H | --help

Display this usage information.

Default: false

Subcommands

The setup command supports the following subcommands:

setup directory-server

Install an OpenDJ directory server instance. See "setup directory-server --help" for specific options.

Options

The setup directory-server command takes the following options:

`-q | --enableStartTls`

Enable StartTLS to allow secure communication with the server using the LDAP port.

Default: false

`-p | --ldapPort {port}`

Port on which the Directory Server should listen for LDAP communication.

`-Z | --ldapsPort {port}`

Port on which the Directory Server should listen for LDAPS communication. The LDAPS port will be configured and SSL will be enabled only if this argument is explicitly specified.

`-a | --addBaseEntry`

Indicates whether to create the base entry in the Directory Server database.

Default: false

`-b | --baseDn {baseDN}`

Base DN for user information in the Directory Server. Multiple base DNs may be provided by using this option multiple times.

`--help-profiles`

Display all available profiles.

Default: false

`--help-profile {name[:version]}`

Display profile parameters.

`-l | --ldifFile {ldifFile}`

Path to an LDIF file containing data that should be added to the Directory Server database. Multiple LDIF files may be provided by using this option multiple times.

`-R | --rejectFile {rejectFile}`

Write rejected entries to the specified file.

-d | --sampleData {numEntries}

Specifies that the database should be populated with the specified number of sample entries.

--skipFile {skipFile}

Write skipped entries to the specified file.

--profile {name[:version]}

Setup profile to apply when initially configuring the server. If the version is not specified, it defaults to the same version as DS. Use this option multiple times to apply multiple profiles. This option cannot be combined with data import options. There are no setup profiles available for this DS version.

--set {[profileName/]parameterName:value}

Assign a value to a setup profile parameter. Setup profile parameters are listed in the parameters.groovy file for the profile. Setup profiles are found in the /tmp/template/setup-profiles directory. Profile name must be provided if multiple profiles are provided. When applying multiple profiles having the same parameter names, indicate the profile that a parameter applies to by using the profileName/parameterName format. Parameter values can contain commons configuration expressions for property value substitution.

-h | --hostname {host}

The fully-qualified directory server host name that will be used when generating self-signed certificates for LDAP SSL/StartTLS, the administration connector, and replication.

Default: localhost.localdomain

--httpPort {port}

Port on which the server should listen for HTTP communication.

--httpsPort {port}

Port on which the server should listen for HTTPS communication.

setup proxy-server

Install an OpenDJ proxy server instance. There are two ways to specify the servers to be contacted by the proxy. They can either be listed exhaustively or retrieved from an existing replication topology. See "setup proxy-server --help" for specific options.

Options

The setup proxy-server command takes the following options:

-q | --enableStartTls

Enable StartTLS to allow secure communication with the server using the LDAP port.

Default: false

-p | --ldapPort {port}

Port on which the Directory Server should listen for LDAP communication.

-Z | --ldapsPort {port}

Port on which the Directory Server should listen for LDAPS communication. The LDAPS port will be configured and SSL will be enabled only if this argument is explicitly specified.

--usePkcs12TrustStore {trustStorePath}

Use existing PKCS12 truststore file to trust the remote server certificates.

--useJceTrustStore {trustStorePath}

Use existing JCEKS truststore file to trust the remote server certificates.

--useJavaTrustStore {trustStorePath}

Use existing JKS truststore file to trust the remote server certificates.

--useJvmTrustStore

Use the JVM truststore for validating remote server certificates.

Default: false

-X | --trustAll

Trust all server SSL certificates.

Default: false

-T | --trustStorePassword {trustStorePassword}

Truststore cleartext password.

-U | --trustStorePasswordFile {path}

Path to a file containing the truststore password.

--loadBalancingAlgorithm {algorithm}

Algorithm to use to load balance between servers. Available algorithms are 'affinity, least-requests'.

Default: affinity

`--staticPrimaryServer {host:port}`

Static server to contact when available before contacting secondary servers. Multiple servers may be provided by using this option multiple times.

`--proxyUserBindDn {proxyBindDN}`

The bind DN for forwarding LDAP requests to remote servers. This bind DN must be present on all the remote servers.

Default: cn=proxy

`--proxyUserBindPassword {proxyBindPassword}`

Password associated with the proxy bind DN. The bind password must be the same on all the remote servers.

`--proxyUserBindPasswordFile {proxyBindPasswordFile}`

Path to a file containing the password associated with the proxy bind DN. The bind password must be the same on all the remote servers.

`--replicationBindDn {bindDN}`

The bind DN for periodically reading replication server configurations. The bind DN must be present on all replication servers and directory servers, it must be able to read the server configuration.

`--replicationBindPassword {bindPassword}`

The bind password for periodically reading replication server configurations. The bind password must be the same on all replication and directory servers.

`--replicationBindPasswordFile {bindPasswordFile}`

Path to a file containing the bind password for periodically reading replication server configurations. The bind password must be the same on all replication and directory servers.

`--replicationPreferredGroupId {domainGroupIDNumber}`

Replication domain group ID number of directory server replicas to contact when available before contacting other replicas. If this option is not specified then all replicas will be treated the same.

`--replicationServer {host:port}`

Replication server to contact periodically in order to discover backend servers. Multiple replication servers may be provided by using this option multiple times.

`--baseDn {baseDN}`

Base DN for user information in the Proxy Server. Multiple base DNs may be provided by using this option multiple times. If no base DNs are defined then the proxy will forward requests to all public naming contexts of the remote servers.

`--staticSecondaryServer {host:port}`

Static server to contact when all primary servers are unavailable. Multiple servers may be provided by using this option multiple times.

`--proxyUsingSsl`

Use SSL to secure communications with remote servers.

Default: false

`--proxyUsingStartTls`

Use Start TLS to secure communication with remote servers.

Default: false

`-h | --hostname {host}`

The fully-qualified directory server host name that will be used when generating self-signed certificates for LDAP SSL/StartTLS, the administration connector, and replication.

Default: localhost.localdomain

`--httpPort {port}`

Port on which the server should listen for HTTP communication.

`--httpsPort {port}`

Port on which the server should listen for HTTPS communication.

setup replication-server

Install OpenDJ as a standalone replication server. The server can be the first of a new replication topology (default behavior) or it can join an existing topology. See "setup replication-server --help" for specific options.

Options

The setup replication-server command takes the following options:

--usePkcs12TrustStore {trustStorePath}

Use existing PKCS12 truststore file to trust certificates from other replication servers in the topology.

--useJceTrustStore {trustStorePath}

Use existing JCEKS truststore file to trust certificates from other replication servers in the topology.

--useJavaTrustStore {trustStorePath}

Use existing JKS truststore file to trust certificates from other replication servers in the topology.

--useJvmTrustStore

Use the JVM truststore to trust certificates from other replication servers in the topology.

Default: false

-X | --trustAll

Trust all server SSL certificates.

Default: false

-T | --trustStorePassword {trustStorePassword}

Truststore cleartext password.

-U | --trustStorePasswordFile {path}

Path to a file containing the truststore password.

--replicationServer {host:port}

Replication server in the topology to be joined. This server must be online during setup. To bind to the remote server, this server uses the global administrator account for the topology. The global administrator account must have ID 'admin', and must use the same password as the root user password for this server.

--replicationPort {port}

Port used for replication protocol communications with other servers.

--secureReplication

Specifies whether the communication through the replication port should be secured. This option is enforced if the --productionMode option is used.

Default: false

-b | --baseDn {baseDN}

Base DN(s) of the data to be replicated. Multiple base DNs can be provided by using this option multiple times. Leave this option empty to replicate all available base DNs in the topology.

-h | --hostname {host}

The fully-qualified directory server host name that will be used when generating self-signed certificates for LDAP SSL/StartTLS, the administration connector, and replication.

Default: localhost.localdomain

--httpPort {port}

Port on which the server should listen for HTTP communication.

--httpsPort {port}

Port on which the server should listen for HTTPS communication.

Exit Codes

0

The command completed successfully.

> 0

An error occurred.

Examples

The following command installs a directory server, enables StartTLS and imports 100 example entries:

```
$ /path/to/opendj/setup directory-server --adminConnectorPort 4444 -b dc=example,dc=com -d 100 \
 -D "cn=Directory Manager" -w password -h opendj.example.com -p 1389 --enableStartTLS


Validating parameters..... Done
Configuring certificates..... Done
Configuring server..... Done
Importing automatically-generated data (100 entries)......... Done
Starting directory server.............. Done

To see basic server status and configuration, you can launch
/path/to/opendj/bin/status
```

Chapter 16.2

upgrade — upgrade OpenDJ configuration and application data

Synopsis

upgrade {options}

Description

Upgrades OpenDJ configuration and application data so that it is compatible with the installed binaries.

This tool should be run immediately after upgrading the OpenDJ binaries and before restarting the server.

NOTE: this tool does not provide backup or restore capabilities. Therefore, it is the responsibility of the OpenDJ administrator to take necessary precautions before performing the upgrade.

This utility performs only part of the upgrade process, which includes the following phases for a single server:

1. Get and unpack a newer version of the software.
2. Stop the current server.
3. Overwrite existing binary and script files with those of the newer version, and then run this utility before restarting the server.
4. Start the upgraded server.

> **Important**
>
> This utility *does not back up your data before you upgrade, nor does it restore your data if the utility fails*. In order to revert a failed upgrade, make sure you back up directory data before you overwrite existing binary and script files.

By default this utility requests confirmation before making important configuration changes. You can use the `--no-prompt` option to run the command non-interactively.

When using the `--no-prompt` option, if this utility cannot complete because it requires confirmation for a potentially very long or critical task, then it exits with an error and a message about how to finish making the changes. You can add the `--force` option to force a non-interactive upgrade to continue in this case, also performing long running and critical tasks.

After upgrading, see the resulting `upgrade.log` file for a full list of operations performed.

Options

The upgrade command takes the following options:

Command options:

`--acceptLicense`

Automatically accepts the product license (if present).

Default: false

`--dataOnly`

Upgrades only application data. OpenDJ configuration must have been upgraded before.

Default: false

`--force`

Forces a non-interactive upgrade to continue even if it requires user interaction. In particular, long running or critical upgrade tasks, such as re-indexing, which require user confirmation will be skipped. This option may only be used with the 'no-prompt' option.

Default: false

`--ignoreErrors`

Ignores any errors which occur during the upgrade. This option should be used with caution and may be useful in automated deployments where potential errors are known in advance and resolved after the upgrade has completed.

Default: false

Utility input/output options:

`-n | --no-prompt`

Use non-interactive mode. If data in the command is missing, the user is not prompted and the tool will fail.

Default: false

`-Q | --quiet`

Use quiet mode.

Default: false

`-v | --verbose`

Use verbose mode.

Default: false

General options:

`-V | --version`

Display Directory Server version information.

Default: false

`-H | --help`

Display this usage information.

Default: false

Exit Codes

0

The command completed successfully.

2

The command was run in non-interactive mode, but could not complete because confirmation was required to run a long or critical task.

See the error message or the log for details.

other

An error occurred.

Appendix A. Getting Support

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see https://www.forgerock.com.

ForgeRock has staff members around the globe who support our international customers and partners. For details on ForgeRock's support offering, including support plans and service level agreements (SLAs), visit https://www.forgerock.com/support.

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

  While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

Glossary

Abandon operation

LDAP operation to stop processing of a request in progress, after which the server drops the connection without a reply to the client application.

Access control

Control to grant or to deny access to a resource.

Access control instruction (ACI)

Instruction added as a directory entry attribute for fine-grained control over what a given user or group member is authorized to do in terms of LDAP operations and access to user data.

ACIs are implemented independently from privileges, which apply to administrative operations.
See also Privilege.

Access control list (ACL)

An access control list connects a user or group of users to one or more security entitlements. For example, users in group sales are granted the entitlement read-only to some financial data.

`access` log

Server log tracing the operations the server processes including timestamps, connection information, and information about the operation itself.

Account lockout

The act of making an account temporarily or permanently inactive after successive authentication failures.

Active user

A user that has the ability to authenticate and use the services, having valid credentials.

Add operation

LDAP operation to add a new entry or entries to the directory.

Anonymous

A user that does not need to authenticate, and is unknown to the system.

Anonymous bind

A bind operation using simple authentication with an empty DN and an empty password, allowing anonymous access such as reading public information.

Approximate index

Index is used to match values that "sound like" those provided in the filter.

Attribute

Properties of a directory entry, stored as one or more key-value pairs. Typical examples include the common name (`cn`) to store the user's full name and variations of the name, user ID (`uid`) to store a unique identifier for the entry, and `mail` to store email addresses.

`audit` log

Type of access log that dumps changes in LDIF.

Authentication

The process of verifying who is requesting access to a resource; the act of confirming the identity of a principal.

Authorization

The process of determining whether access should be granted to an individual based on information about that individual; the act of determining whether to grant or to deny a principal access to a resource.

Backend

Repository that stores directory data. Different implementations with different capabilities exist.

Binary copy

Binary backup archive of one directory server that can be restored on another directory server.

Bind operation

LDAP authentication operation to determine the client's identity in LDAP terms, the identity which is later used by the server to authorize (or not) access to directory data that the client wants to lookup or change.

Branch

The distinguished name (DN) of a non-leaf entry in the Directory Information Tree (DIT), and also that entry and all its subordinates taken together.

Some administrative operations allow you to include or exclude branches by specifying the DN of the branch.

See also Suffix.

Collective attribute

A standard mechanism for defining attributes that appear on all the entries in a particular subtree.

Compare operation

LDAP operation to compare a specified attribute value with the value stored on an entry in the directory.

Control

Information added to an LDAP message to further specify how an LDAP operation should be processed. DS supports many LDAP controls.

Database cache

Memory space set aside to hold database content.

`debug` log

Server log tracing details needed to troubleshoot a problem in the server.

Delete operation

LDAP operation to remove an existing entry or entries from the directory.

Directory

A directory is a network service which lists participants in the network such as users, computers, printers, and groups. The directory provides a convenient, centralized, and robust mechanism for publishing and consuming information about network participants.

Directory hierarchy

A directory can be organized into a hierarchy in order to make it easier to browse or manage. Directory hierarchies normally represent something in the physical world, such as organizational hierarchies or physical locations. For example, the top level of a directory may represent a company, the next level down divisions, the next level down departments, and down the hierarchy. Alternately, the top level may represent the world, the next level down countries, next states or provinces, and next cities.

Directory Information Tree (DIT)

A set of directory entries organized hierarchically in a tree structure, where the vertices are the entries and the arcs between vertices define relationships between entries

Directory manager

Default directory superuser who has privileges to do full administration of the DS server, including bypassing access control evaluation, changing access controls, and changing administrative privileges.
See also Superuser.

Directory object

A directory object is an item in a directory. Example objects include users, user groups, computers, and more. Objects may be organized into a hierarchy and contain identifying attributes.
See also Entry.

Directory proxy server

Server that forwards LDAP requests to remote directory servers. A standalone directory proxy server does not store user data.
See also Directory server.

Directory server

Server application for centralizing information about network participants. A highly available directory service consists of multiple directory servers configured to replicate directory data.
See also Directory, Replication.

Directory Services Markup Language (DSML)

Standard language to access directory services using XML. DMSL v1 defined an XML mapping of LDAP objects, while DSMLv2 maps the LDAP Protocol and data model to XML.

Distinguished name (DN)

Fully qualified name for a directory entry, such as `uid=bjensen,ou=People,dc=example,dc=com`, built by concatenating the entry RDN (`uid=bjensen`) with the DN of the parent entry (`ou=People,dc=example,dc=com`).

Domain

A replication domain consists of several directory servers sharing the same synchronized set of data.

The base DN of a replication domain specifies the base DN of the replicated data.

DSML gateway

Standalone web application that translates DSML requests from client applications to LDAP requests to a directory service, and LDAP responses from a directory service to DSML responses to client applications.

Dynamic group

Group that specifies members using LDAP URLs.

Entry

As generic and hierarchical data stores, directories always contain different kinds of entries, either nodes (or containers) or leaf entries. An entry is an object in the directory, defined by one of more object classes and their related attributes. At startup, DS servers report the number of entries contained in each suffix.

Entry cache

Memory space set aside to hold frequently accessed, large entries, such as static groups.

Equality index

Index used to match values that correspond exactly (though generally without case sensitivity) to the value provided in the search filter.

`errors` log

Server log tracing server events, error conditions, and warnings, categorized and identified by severity.

Export

Save directory data in an LDIF file.

Extended operation

Additional LDAP operation not included in the original standards. DS servers support several standard LDAP extended operations.

Extensible match index

Index for a matching rule other than approximate, equality, ordering, presence, substring or VLV, such as an index for generalized time.

External user

An individual that accesses company resources or services but is not working for the company. Typically a customer or partner.

Etime

Elapsed time within the server to process a request, starting from the moment the decoded operation is available to be processed by a worker thread.

Filter

An LDAP search filter is an expression that the server uses to find entries that match a search request, such as `(mail=*@example.com)` to match all entries having an email address in the example.com domain.

Group

Entry identifying a set of members whose entries are also in the directory.

Idle time limit

Defines how long DS allows idle connections to remain open.

Import

Read in and index directory data from an LDIF file.

Inactive user

An entry in the directory that once represented a user but which is now no longer able to be authenticated.

Index

Directory server backend feature to allow quick lookup of entries based on their attribute values.
See also Approximate index, Equality index, Extensible match index, Ordering index, Presence index, Substring index, Virtual list view (VLV) index, Index entry limit.

Index entry limit

When the number of entries that an index key points to exceeds the index entry limit, DS stops maintaining the list of entries for that index key.

Internal user

An individual who works within the company either as an employee or as a contractor.

LDAP Data Interchange Format (LDIF)

Standard, portable, text-based representation of directory content. See RFC 2849.

LDAP URL

LDAP Uniform Resource Locator such as `ldap://directory.example.com:389/dc=example,dc=com??sub?(uid=bjensen)`. See RFC 2255.

LDAPS

LDAP over SSL.

Lightweight Directory Access Protocol (LDAP)

A simple and standardized network protocol used by applications to connect to a directory, search for objects and add, edit or remove objects. See RFC 4510.

Lookthrough limit

Defines the maximum number of candidate entries DS considers when processing a search.

Matching rule

Defines rules for performing matching operations against assertion values. Matching rules are frequently associated with an attribute syntax and are used to compare values according to that syntax. For example, the `distinguishedNameEqualityMatch` matching rule can be used to determine whether two DNs are equal and can ignore unnecessary spaces around commas and equal signs, differences in capitalization in attribute names, and other discrepancies.

Modify DN operation

LDAP modification operation to request that the server change the distinguished name of an entry.

Modify operation

LDAP modification operation to request that the server change one or more attributes of an entry.

Naming context

Base DN under which client applications can look for user data.

Object class

Identifies entries that share certain characteristics. Most commonly, an entry's object classes define the attributes that must and may be present on the entry. Object classes are stored on entries as values of the `objectClass` attribute. Object classes are defined in the directory schema, and can be abstract (defining characteristics for other object classes to inherit), structural (defining the basic structure of an entry, one structural inheritance per entry), or auxiliary (for decorating entries already having a structural object class with other required and optional attributes).

Object identifier (OID)

String that uniquely identifies an object, such as `0.9.2342.19200300.100.1.1` for the user ID attribute or `1.3.6.1.4.1.1466.115.121.1.15` for `DirectoryString` syntax.

Operational attribute

An attribute that has a special (operational) meaning for the server, such as `pwdPolicySubentry` or `modifyTimestamp`.

Ordering index

Index used to match values for a filter that specifies a range.

Password policy

A set of rules regarding what sequence of characters constitutes an acceptable password. Acceptable passwords are generally those that would be too difficult for another user or an automated program to guess and thereby defeat the password mechanism. Password policies may require a minimum length, a mixture of different types of characters (lowercase, uppercase, digits, punctuation marks, and other characters), avoiding dictionary words or passwords based on the user's name, and other attributes. Password policies may also require that users not reuse old passwords and that users change their passwords regularly.

Password reset

Password change performed by a user other than the user who owns the entry.

Password storage scheme

Mechanism for encoding user passwords stored on directory entries. DS implements a number of password storage schemes.

Password validator

Mechanism for determining whether a proposed password is acceptable for use. DS implements a number of password validators.

Plugin

Java library with accompanying configuration that implements a feature through processing that is not essential to the core operation of DS servers.

As the name indicates, plugins can be plugged in to an installed server for immediate configuration and use without recompiling the server.

DS servers invoke plugins at specific points in the lifecycle of a client request. The DS configuration framework lets directory administrators manage plugins with the same tools used to manage the server.

Presence index

Index used to match the fact that an attribute is present on the entry, regardless of the value.

Principal

Entity that can be authenticated, such as a user, a device, or an application.

Privilege

Server configuration settings controlling access to administrative operations such as exporting and importing data, restarting the server, performing password reset, and changing the server configuration.

Privileges are implemented independently from access control instructions (ACI), which apply to LDAP operations and user data. See also Access control instruction (ACI).

Referential integrity

Ensuring that group membership remains consistent following changes to member entries.

`referint` log

Server log tracing referential integrity events, with entries similar to the errors log.

Referral

Reference to another directory location, which can be another directory server running elsewhere or another container on the same server, where the current operation can be processed.

Relative distinguished name (RDN)

Initial portion of a DN that distinguishes the entry from all other entries at the same level, such as `uid=bjensen` in `uid=bjensen,ou=People,dc=example,dc=com`.

Replica

Directory server this is configured to use replication.

Replication

Data synchronization that ensures all directory servers participating eventually share a consistent set of directory data.

`replication` log

Server log tracing replication events, with entries similar to the errors log.

Replication server

Server dedicated to transmitting replication messages. A standalone replication server does not store user data.

REST to LDAP gateway

Standalone web application that translates RESTful HTTP requests from client applications to LDAP requests to directory services, and

LDAP responses from directory services to HTTP responses to client applications.

Root DSE

The directory entry with distinguished name "" (empty string), where DSE is an acronym for DSA-Specific Entry. DSA is an acronym for Directory Server Agent, a single directory server. The root DSE serves to expose information over LDAP about what the directory server supports in terms of LDAP controls, auth password schemes, SASL mechanisms, LDAP protocol versions, naming contexts, features, LDAP extended operations, and other information.

Schema

LDAP schema defines the object classes, attributes types, attribute value syntaxes, matching rules and other constrains on entries held by the directory server.

Search filter

See Filter.

Search operation

LDAP lookup operation where a client requests that the server return entries based on an LDAP filter and a base DN under which to search.

Simple authentication

Bind operation performed with a user's entry DN and user's password. Use simple authentication only if the network connection is secure.

Size limit

Sets the maximum number of entries returned for a search.

Static group

Group that enumerates member entries.

Subentry

An entry, such as a password policy entry, that resides with the user data but holds operational data, and is not visible in search results unless explicitly requested.

Substring index

Index used to match values specified with wildcards in the filter.

Suffix

The distinguished name (DN) of a root entry in the Directory Information Tree (DIT), and also that entry and all its subordinates taken together as a single object of administrative tasks such as export, import, indexing, and replication.

Superuser

User with privileges to perform unconstrained administrative actions on DS server. This account is analogous to the UNIX `root` and Windows `Administrator` accounts.

Superuser privileges include the following:

- `bypass-acl`: The holder is not subject to access control.
- `privilege-change`: The holder can edit administrative privileges.
- `proxied-auth`: The holder can make requests on behalf of another user, including directory superusers.

The default superuser is `cn=Directory Manager`. You can create additional superuser accounts, each with different administrative privileges. See also Directory manager, Privilege.

Task

Mechanism to provide remote access to server administrative functions. DS software supports tasks to back up and restore backends, to import and export LDIF files, and to stop and restart the server.

Time limit

Defines the maximum processing time DS devotes to a search operation.

Unbind operation

LDAP operation to release resources at the end of a session.

Unindexed search

Search operation for which no matching index is available. If no indexes are applicable, then the directory server potentially has to go through all entries to look for candidate matches. For this reason, the `unindexed-search` privilege, which allows users to request searches for which no applicable index exists, is reserved for the directory manager by default.

User

An entry that represents an individual that can be authenticated through credentials contained or referenced by its attributes. A user may represent an internal user or an external user, and may be an active user or an inactive user.

User attribute

An attribute for storing user data on a directory entry such as `mail` or `givenname`.

Virtual attribute

An attribute with dynamically generated values that appear in entries but are not persistently stored in the backend.

Virtual directory

An application that exposes a consolidated view of multiple physical directories over an LDAP interface. Consumers of the directory information connect to the virtual directory's LDAP service. Behind the scenes, requests for information and updates to the directory are sent to one or more physical directories where the actual information resides. Virtual directories enable organizations to create a consolidated view of information that for legal or technical reasons cannot be consolidated into a single physical copy.

Virtual list view (VLV) index

Browsing index designed to help the directory server respond to client applications that need, for example, to browse through a long list of results a page at a time in a GUI.

Virtual static group

DS group that lets applications see dynamic groups as what appear to be static groups.

X.500

A family of standardized protocols for accessing, browsing and maintaining a directory. X.500 is functionally similar to LDAP, but is generally considered to be more complex, and has consequently not been widely adopted.